Package: PoolBal (via r-universe)

October 18, 2024

Version 0.1-0

Encoding UTF-8

Title Balancing Central and Marginal Rejection of Pooled p-Values

Description When using pooled p-values to adjust for multiple testing, there is an inherent balance that must be struck between rejection based on weak evidence spread among many tests and strong evidence in a few, explored in Salahub and Olford (2023) <arXiv:2310.16600>. This package provides functionality to compute marginal and central rejection levels and the centrality quotient for p-value pooling functions and provides implementations of the chi-squared quantile pooled p-value (described in Salahub and Oldford (2023)) and a proposal from Heard and Rubin-Delanchy (2018) <doi:10.1093/biomet/asx076> to control the quotient's value.

Author Chris Salahub [aut, cre]

Maintainer Chris Salahub <`chris.salahub@uwaterloo.ca`>

Depends R (>= 4.3.0)

Imports methods

License GPL (>= 3)

NeedsCompilation no

Repository CRAN

RoxygenNote 7.2.3

Date/Publication 2023-11-22 10:10:02 UTC

Contents

`altFrequencyMat` *Identify a region of plausible alternative hypotheses in the proportion, strength of non-null evidence space*

Description

This function provides a convenient way to interact with simulations performed over a grid of possible alternatives spanning the proportion (eta) and strength (KL divergence) of evidence against the null hypothesis under beta alternatives.

Usage

```
altFrequencyMat(logKappaRange, logW = FALSE)
```

Arguments

`logKappaRange`	pair of numeric values
`logW`	logical, should the log scale simulation be used?

Details

The simulation this function summarized used a range of eta, w, and KL divergence values to generate thousands of potential alternative distributions. The power of each chi-squared pooled p-value for 161 kappa values ranging from exp(-8) to exp(8) selected uniformly on the log scale was then computed for each alternative using 10,000 simulated examples. Every choice of kappa was compared to the maximum power across all kappas for each setting using a binomial test of differences. This same simulation was repeated twice: once for w values selected uniformly from 0 to 1 and another where selection was uniform on the log scale. The internal data summarizes the results by reporting the count of instances in w (or logw) where a given kappa value was most powerful for a given eta and KL divergence.

Though the simulation data is not exported to users and so cannot be accessed directly, this function allows a user to query the data with a range of kappa values (corresponding to those where a given sample seems most powerful) and returns the count of cases in w where a kappa in the corresponding kappa range was most powerful given the eta, KL-divergence combination with beta alternatives. The simulations only spanned kappa values from exp(-8) to exp(8), so providing values outside this range will give very inaccurate results.

Value

An 81 by 81 matrix giving summarized counts of cases.

Author(s)

Chris Salahub

Examples

```
altFrequencyMat(c(-1, 1), logW = FALSE)
altFrequencyMat(c(-1, 1), logW = TRUE)
```

---

betaDiv — *Compute the Kullback-Leibler divergence between the beta and uniform distributions*

Description

Computes the Kullback-Leibler divergence for the special case of the uniform density against the beta density.

Usage

```
betaDiv(a, w = (1 - a)/(b - a), b = 1/w + a * (1 - 1/w))
```

Arguments

a	first shape parameter between 0 and infinity
w	UMP parameter between 0 and 1
b	second shape parameter between 0 and infinity

Details

This function accepts either the a/b parameterization (equivalent to shape1/shape2 in R), or the a/w parameterization which links the divergence to the UMP test.

Value

A real value.

Author(s)

Chris Salahub

Examples

```
betaDiv(a = 0.5, w = 0.5)
betaDiv(a = 0.1, b = 1)
```

---

chiKappa *Chi-squared kappa for a given centrality quotient*

Description

Computes the kappa (degrees of freedom) required to obtain a given centrality quotient using the chi-square pooled p-value.

Usage

```
chiKappa(
  cq,
  M,
  alpha = 0.05,
  interval = c(0, 100),
  tol = .Machine$double.eps^0.5
)
```

Arguments

`cq`	numeric between 0 and 1
`M`	integer sample size greater than 0
`alpha`	numeric between 0 and 1
`interval`	numeric of length 2, where should roots be sought?
`tol`	numeric, how close do values need to be for equality?

Details

This function is essentially a wrapper for uniroot which finds where chiCentQuot gives an output equal to the given centrality quotient to provide an approximate kappa giving that quotient.

Value

A numeric within interval.

Author(s)

Chris Salahub

Examples

```
chiKappa(0.5, 10, 0.05)
chiKappa(0.5, 20, 0.05)
chiKappa(0.5, 100, 0.05, interval = c(0, 10))
```

---

chiPc *Chi-squared central rejection level*

Description

Computes the central rejection level for the chi-squared pooled p-value.

Usage

```
chiPc(kappa, M, alpha = 0.05)
```

Arguments

kappa	numeric between 0 and infinity
M	integer sample size greater than 0
alpha	numeric between 0 and 1

Details

The central rejection level is the maximum p-value shared among all tests which still results in rejection of the null using a pooled p-value. For the chi-squared pooled p-value, this is an upper tail probability of the chi-squared distribution. This function computes the upper tail probability for a given sample size M, degrees of freedom kappa, and rejection level alpha.

Value

A numeric between 0 and 1.

Author(s)

Chris Salahub

Examples

```
chiPc(2, 10, 0.05)
chiPc(2, 20, 0.05) # increases in sample size
```

`chiPool` *Chi-squared p-value pooling*

Description

This implements the chi-squared pooled p-value which can be used to control the centrality quotient when pooling p-values.

Usage

```
chiPool(p, kappa)
```

Arguments

`p`	numeric vector of p-values between 0 and 1
`kappa`	numeric value between 0 and infinity

Details

The chi-squared pooled p-value is a quantile transformation pooled p-value based on the chi-squared distribution with degrees of freedom kappa. By setting kappa between 0 and infinity, smooth interpolation is achieved between Tippett's minimum pooled p-value and Stouffer's normal quantile pooled p-value respectively. Choosing a kappa value of 2, Fisher's pooling function is obtained. Tippett's pooled p-value is maximally non-central and Stouffer's is maximally central, while Fisher's presents a balance between marginal and central rejection.

Value

A pooled p-value between 0 and 1.

Author(s)

Chris Salahub

Examples

```
p <- c(0.1, 0.5, 0.9)
chiPool(p, exp(-4))
chiPool(p, 2)
chiPool(p, exp(4))
```

`chiPr` *Chi-squared marginal rejection level*

Description

Computes the marginal rejection level for the chi-squared pooled p-value.

Usage

```
chiPr(kappa, M, alpha = 0.05)
```

Arguments

`kappa`	numeric between 0 and infinity
`M`	integer sample size greater than 0
`alpha`	numeric between 0 and 1

Details

The marginal rejection level is the maximum p-value in a single test which results in rejection when all other tests produce p-values of one. For the chi-squared pooled p-value, this is an upper tail probability of the chi-squared distribution. This function computes the upper tail probability for a given sample size M, degrees of freedom kappa, and rejection level alpha.

Value

A numeric between 0 and 1.

Author(s)

Chris Salahub

Examples

```
chiPr(2, 10, 0.05)
chiPr(2, 20, 0.05)
```

`chiQ` *Chi-squared centrality quotient*

Description

Computes the centrality quotient of the chi-square pooled p-value.

Usage

```
chiQ(kappa, M, alpha = 0.05)
```

Arguments

`kappa`	numeric between 0 and infinity
`M`	integer sample size greater than 0
`alpha`	numeric between 0 and 1

Details

The centrality quotient of a pooled p-value measures the relative preference it gives to p-values all sharing the same level of evidence over a single test with strong evidence relative to others. For the chi-square pooled p-value, this is a conditional probability which this function computes.

Value

A numeric between 0 and 1.

Author(s)

Chris Salahub

Examples

```
chiQ(2, 10, 0.05)
chiQ(2, 20, 0.05)
chiQ(0.5, 20, 0.05)
```

convertGeneticSigma *Convert p-value correlation to chi-squared covariance*

Description

Convert a matrix of correlations between p-values to a matrix of covariances between their chi-squared transforms.

Usage

```
convertGeneticSigma(sigma, kappa, models = chiCorMods)
```

Arguments

`sigma`	M by M correlation matrix between markers
`kappa`	numeric degrees of freedom
`models`	model object with a predict method

Details

This function uses models fit to large simulated data sets to convert a matrix of correlations between genetic markers the covariance matrix of chi-squared random variables gained from transforming p-values on these markers. The simulations used to create data for these models assume the p-values for each marker arise from tests of association with a common, normally distributed trait independent of all markers. As a result, this conversion function should be used only in analogous settings.

Models were fit for degrees of freedom at increments of 0.1 between -8 and 8 on the log scale, and interpolation is applied if the degrees of freedom given to the function does not fall exactly on this grid (with a warning provided to the user).

If a user wants to generalize this setting, the option to provide a custom list of models which predict based on a named argument ‘zcor‘ is supported. Each model must have a name in the list that can be converted to a numeric, and these are assumed to be on the natural log scale.

Value

M by M matrix of chi-squared covariances

Author(s)

Chirs Salahub

estimatePc *Compute the central rejection level*

Description

Estimates the central rejection level for an arbitrary pooled p-value function.

Usage

```
estimatePc(
  poolFun,
  alpha = 0.05,
  M = 2,
  interval = c(0, 1),
  poolArgs = list(),
  ...
)
```

Arguments

`poolFun`	function accepting a vector of p-values
`alpha`	numeric between 0 and 1
`M`	integer, how many p-values are there?
`interval`	two numerics giving the bounds of root-searching
`poolArgs`	(optional) additional named arguments for poolFun
`...`	additional arguments to uniroot

Details

The central rejection level is the maximum p-value shared among all tests which still results in rejection of the null using a pooled p-value.

This function is essentially a wrapper for uniroot, and accepts a pooling function which takes a numeric vector as its first argument and potentially other arguments given in poolArgs and returns a single value. Using this pooling function, a specified dimension M and a rejection level alpha, uniroot searches for the root to poolFun - alpha along the line where all p-values are equal.

Value

The uniroot output.

Author(s)

Chris Salahub

Examples

```
tippool <- function(p) 1 - (1 - min(p))^(length(p))
estimatePc(tippool, 0.05, M = 10, interval = c(0, 1))
```

`estimatePrb` *Compute the marginal rejection level*

Description

Estimates the marginal rejection level for an arbitrary pooled p-value function.

Usage

```
estimatePrb(
  poolFun,
  alpha = 0.05,
  b = 1,
  M = 2,
  interval = c(0, b),
  poolArgs = list(),
  ...
)
```

Arguments

`poolFun`	function accepting a vector of p-values
`alpha`	numeric between 0 and 1
`b`	numeric, the value of the M - 1 repeated p-values
`M`	integer, how many p-values are there?
`interval`	two numerics giving the bounds of root-searching
`poolArgs`	(optional) additional named arguments for poolFun
`...`	additional arguments to uniroot

Details

The marginal rejection level is the maximum p-value in a single test less than b which still results in rejection of the null when all other tests have a p-value of b.

This function is essentially a wrapper for uniroot, and accepts a pooling function which takes a numeric vector as its first argument and potentially other arguments given in poolArgs and returns a single value. Using this pooling function, a specified dimension M and a rejection level alpha, uniroot searches for the root to poolFun - alpha along one margin when all other p-values are equal to b.

Value

The uniroot output.

Author(s)

Chris Salahub

Examples

```
stopool <- function(p) pnorm(sum(qnorm(p, lower.tail = FALSE))/ sqrt(length(p)), lower.tail = FALSE)
estimatePrb(stopool, 0.05, M = 10, interval = c(.Machine$double.eps, 1))
estimatePrb(stopool, 0.05, M = 10, b = 0.5, interval = c(.Machine$double.eps, 1))
```

`estimateQ` *Compute the centrality quotient*

Description

Estimates the centrality quotient for an arbitrary pooled p-value function.

Usage

```
estimateQ(
  poolFun,
  alpha = 0.05,
  M = 2,
  interval = c(0, 1),
  poolArgs = list(),
  ...
)
```

Arguments

`poolFun`	function accepting a vector of p-values
`alpha`	numeric between 0 and 1
`M`	integer, how many p-values are there?
`interval`	two numerics giving the bounds of root-searching
`poolArgs`	(optional) additional named arguments for poolFun
`...`	additional arguments to uniroot

Details

The centrality quotient communicates the tendency for a test to favour evidence shared among all tests over strong evidence in a single test.

This function uses the individual estimation functions for central and marginal rejection levels to compute the centrality quotient for an arbitrary pooled p-value function. The option to specify b for marginal rejection is included in case the pooled p -value has strange behaviour when p-values are equal to 1.

Value

The uniroot output.

Author(s)

Chris Salahub

Examples

```
estimateQ(chiPool, alpha = 0.05, M = 10, poolArgs = list(kappa = 10))
```

`findA` *Estimate parameter for a given beta KL divergence and UMP test*

Description

Computes the first parameter value for a given KL divergence and UMP test.

Usage

```
findA(w, logd = 0, ...)
```

Arguments

`w`	UMP parameter between 0 and 1
`logd`	numeric value, the log KL divergence
`...`	additional arguments to uniroot

Details

This function uses uniroot to invert the beta divergence for a given w and return the a value which gives that beta divergence given the UMP parameter w. The search interval is specified internally, so should not be passed in using additional argument.

Value

A real value.

Author(s)

Chris Salahub

Examples

```
findA(0.5, logd = 0)
```

hrPc — *Empirical UMP beta central rejection level*

Description

Uses simulation to estimate the central rejection level for the UMP pooled p-value of a restricted beta family

Usage

```
hrPc(w, alpha = 0.05, M = 2, nsim = 1e+05)
```

Arguments

`w`	numeric between 0 and 1
`alpha`	numeric between 0 and 1
`M`	integer sample size greater than 0
`nsim`	integer, the number of simulated null cases generated

Details

The central rejection level is the maximum p-value shared among all tests which still results in rejection of the null using a pooled p-value.

To test the null hypotheses that all p-values are uniform against a restricted beta family 0 < a <= 1 <= b, the most powerful pooled p-value linearly combines upper and lower tail probabilities of the chi-squared distribution with two degrees of freedom with weights w and (1 - w) where w = (1 - a)/(b - a).

This function estimates the central rejection level empirically by simulating a specified number of null cases to give an empirical pooled p-value for the rejection level alpha.

Value

A numeric between 0 and 1.

Author(s)

Chris Salahub

Examples

```
hrPc(w = 0.5, alpha = 0.05, M = 10)
hrPc(w = 0.5, alpha = 0.05, M = 20)
```

`hrPool` *Empirical UMP beta pooled p-value*

Description

Uses simulation under the null to approximate the UMP pooled p-value for a restricted beta family.

Usage

```
hrPool(w = 1, M = 10, nsim = 1e+05)
```

Arguments

`w`	numeric value between 0 and 1
`M`	integer, the number of tests to pool
`nsim`	integer, the number of simulated null cases generated

Details

To test the null hypotheses that all p-values are uniform against a restricted beta family 0 < a <= 1 <= b, the most powerful pooled p-value linearly combines upper and lower tail probabilities of the chi-squared distribution with two degrees of freedom with weights w and (1 - w) where w = (1 - a)/(b - a).

This function computes the statistic given by this combination for a collection of p-values, and then simulates a specified number of null cases to give an empirical pooled p-value. It produces a closure so that the time-intensive simulation step doesn't need to be repeated.

Value

A closure which accepts a vector of values between 0 and 1 and returns a single numeric between 0 and 1

Author(s)

Chris Salahub

Examples

```
p <- c(0.1, 0.5, 0.9)
hr2 <- hrPool(w = 0.2, M = 3)
hr2(p)
hr5 <- hrPool(w = 0.5, M = 3, nsim = 100)
hr5(p)
```

`hrPr` *Empirical UMP beta marginal rejection level*

Description

Uses simulation to estimate the marginal rejection level for the UMP pooled p-value of a restricted beta family

Usage

```
hrPr(w, alpha = 0.05, M = 2, nsim = 1e+05)
```

Arguments

`w`	numeric between 0 and 1
`alpha`	numeric between 0 and 1
`M`	integer sample size greater than 0
`nsim`	integer, the number of simulated null cases generated

Details

The marginal rejection level is the maximum p-value in a single tests which still results in rejection of the null when all other tests have a p-value of 1.

To test the null hypotheses that all p-values are uniform against a restricted beta family 0 < a <= 1 <= b, the most powerful pooled p-value linearly combines upper and lower tail probabilities of the chi-squared distribution with two degrees of freedom with weights w and (1 - w) where w = (1 - a)/(b - a).

This function estimates the marginal rejection level empirically by simulating a specified number of null cases to give an empirical pooled p-value for the rejection level alpha.

Value

A numeric between 0 and 1.

Author(s)

Chris Salahub

Examples

```
hrPr(w = 0.5, alpha = 0.05, M = 10)
hrPr(w = 0.5, alpha = 0.05, M = 10) # decreases in sample size
```

hrQ *Empirical UMP beta centrality quotient*

Description

Estimates the centrality quotient for the UMP pooled p-value of a restricted beta family.

Usage

```
hrQ(w, alpha = 0.05, M = 2, nsim = 1e+05)
```

Arguments

`w`	numeric between 0 and 1
`alpha`	numeric between 0 and 1
`M`	integer sample size greater than 0
`nsim`	integer, the number of simulated null cases generated

Details

The centrality quotient communicates the tendency for a test to favour evidence shared among all tests over strong evidence in a single test.

To test the null hypotheses that all p-values are uniform against a restricted beta family 0 < a <= 1 <= b, the most powerful pooled p-value linearly combines upper and lower tail probabilities of the chi-squared distribution with two degrees of freedom with weights w and (1 - w) where w = (1 - a)/(b - a).

This function uses the individual estimation functions for central and marginal rejection levels to compute the centrality quotient for the UMP pooled p-value.

Value

An empirical estimate of the centrality quotient.

Author(s)

Chris Salahub

Examples

```
hrQ(0.8, alpha = 0.05, M = 10)
```

hrStat *UMP beta p-value pooled statistic*

Description

Computes the UMP p-value pooling statistic for a restricted beta family.

Usage

```
hrStat(p, w = 1)
```

Arguments

`p`	numeric vector of p-values between 0 and 1
`w`	numeric value between 0 and 1

Details

To test the null hypotheses that all p-values are uniform against a restricted beta family 0 < a <= 1 <= b, the most powerful pooled p-value linearly combines upper and lower tail probabilities of the chi-squared distribution with two degrees of freedom with weights w and (1 - w) where w = (1 - a)/(b - a).

This function computes the statistic given by this combination for a collection of p-values, simulation or approximation is required to convert this to a p-value.

Value

A numeric value giving the pooled statistic.

Author(s)

Chris Salahub

Examples

```
p <- c(0.1, 0.5, 0.9)
hrStat(p, 0.2)
hrStat(p, 0.5)
hrStat(p, 0.9)
```

klDiv *Compute the Kullback-Leibler divergence*

Description

Computes the Kullback-Leibler divergence for two arbitrary densities f1 and f2.

Usage

```
klDiv(f1, f2, lower = 0, upper = 1)
```

Arguments

`f1`	density function of a real-valued random variable
`f2`	density function of a real-values random variable
`lower`	real value, the lower bound of integration
`upper`	real value, the upper bound of integration

Details

Given lower and upper bounds, this function integrates the expression for the Kullback-Leibler divergence KL(f1|f2).

Value

A real value.

Author(s)

Chris Salahub

Examples

```
klDiv(dunif, function(x) dbeta(x, 0.5, 1))
```

marHistHeatMap *Heatmap with marginal histograms*

Description

Display a matrix using a heatmap with marginal histograms.

Usage

```
marHistHeatMap(
  mat,
  main = "",
  ylab = expression(eta),
  xlab = "lnD(a,w)",
  pal = NULL,
  histFill = adjustcolor("firebrick", 0.5),
  ...
)
```

Arguments

`mat`	numeric matrix to be plotted
`main`	title
`ylab`	y axis label
`xlab`	x axis label
`pal`	palette for heatmap
`histFill`	colour to fill histogram bars
`...`	additional arguments to image

Details

This function accepts a matrix of values and plots the matrix with saturation/hue determined by a provided palette argument generated by colorRampPalette, for example. Marginal histograms summarizing the relative frequencies along both dimensions are also plotted to give a complete sense of the individual distributions alongside their joint distribution. This was designed to summarize the alternative distribution space summarized by altFrequencyMat, and the defaults reflect this.

Value

Plot the data using a heatmap and marginal histograms and return nothing.

Author(s)

Chris Salahub

Examples

```
marHistHeatMap(altFrequencyMat(c(0, 2)))
```

`rBetaH4` *Generate realizations of beta alternative distributions*

Description

These functions can be used to generate samples of p-values all following a beta distribution (H4) or following either uniform or beta distributions according to proportion eta (H3).

Usage

```
rBetaH4(a, b = 1/w + a * (1 - 1/w), w = (1 - a)/(b - a), M = 2, N = 10)

rBetaH3(
  a,
  b = 1/w + a * (1 - 1/w),
  w = (1 - a)/(b - a),
  eta = 0.5,
  M = 2,
  N = 10
)
```

Arguments

`a`	first beta parameter, numeric between 0 and infinity
`b`	second beta parameter, numeric between 0 and infinity
`w`	UMP parameter between 0 and 1
`M`	number of p-values per realization
`N`	number of realizations
`eta`	numeric between 0 and 1, proportion of non-null tests per sample

Details

Alternatives

These functions are provided as a convenience, and support a/b (shape1/shape2) or a/w specification of beta parameters.

Value

An N by M matrix of simulated p-values.

Functions

- `rBetaH4()`: iid Beta(a,w) p-values
- `rBetaH3()`: M*eta iid Beta(a,w) p-values, others uniform

Author(s)

Chris Salahub

Examples

```
rBetaH4(a = 0.5, b = 1.5, M = 10, N = 100)
rBetaH3(a = 0.5, b = 1.5, eta = 0.5, M = 10, N = 100)
```

`satterApproxP` *Satterthwaite p-values*

Description

p-value of the sum of dependent chi-squared using the Satterthwaite approximation for the degrees of freedom.

Usage

```
satterApproxP(qs, covmat, kappa)
```

Arguments

`qs`	M numeric values (observed chi-squared values)
`covmat`	M by M covariance matrix of qs
`kappa`	degrees of freedom of qs

Details

Computes the p-value of an observed vector of chi-squared variables using the Satterthwaite approximation. This approximates the sum of dependent chi-squared variables with a scaled chi-squared distribution with degrees of freedom chosen to match the first two moments of the dependent sum.

Value

a numeric in [0,1], the p-value of the sum

Author(s)

Chris Salahub

`satterChiPool` *Pool p-values using the Satterthwaite approximation*

Description

Compute the pooled p-value of dependent p-values based on the dependence present when they are all converted to chi-squared random variables by the same chi-squared quantile function.

Usage

```
satterChiPool(ps, covmat, kappa)
```

Arguments

`ps`	numeric vector of M p-values
`covmat`	M by M covariance matrix of chi-squared random variables arising from quantile transformations of ps
`kappa`	numeric degrees of freedom

Details

Care must be taken in the arguments for this function, as the covmat argument accepts the covariance of the transformed variables rather than the covariance of the p-values, and so passes the argument covmat directly to the function that computes the Satterthwaite approximation. For the case of genetic markers, the ‘convertGeneticSigma‘ function provides the appropriate matrix given a genetic correlation matrix.

Value

A pooled p-value between 0 and 1.

Author(s)

Chris Salahub

Index