Package 'DBpower'

February 10, 2022

Type Package

Title Finite Sample Power Calculations for Detection Boundary Tests

Version 0.1.0

Description Calculates lower bound on power, upper bound on power, and exact power (small sets only) for detection boundary tests (e.g. Berk-Jones, Generalized Berk-Jones, innovated Berk-Jones) used in set-based inference studies. These detection boundary tests are described in Sun et al., (2019) <doi:10.1080/01621459.2019.1660170>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Imports dplyr, magrittr, stats, mvtnorm, combinat, GBJ, kit,

Suggests knitr, bindata, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Ryan Sun [aut, cre]

Maintainer Ryan Sun <ryansun.work@gmail.com>

Repository CRAN

Date/Publication 2022-02-10 19:10:05 UTC

R topics documented:

`calcb1` *calc_b1.R*

Description

Calculate lower bound or upper bound on power when considering only the largest test statistic in magnitude, i.e. only |Z|_(J) and not |Z|_(J-1).

Usage

```
calcb1(lower = TRUE, upper = FALSE, muVec, sigMat, bounds)
```

Arguments

`lower`	Boolean, whether to calculate lower bound.
`upper`	Boolean, whether to calculate upper bound.
`muVec`	Mean vector of test statistics under the alternative (assuming it's MVN).
`sigMat`	Covariance matrix of test statistics under the alternative (assuming it's MVN).
`bounds`	A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).

Value

A list with the elements:

`allProbsLower`	J*1 vector of all components summed to calculate lower bound.
`lowerProb`	Lower bound.
`allProbsUpper`	J*1 vector of all components summed to calculate upper bound.
`upperProb`	Upper bound.

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
myBounds <- set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
calcb1(muVec = c(1, 0, 0, 0, 0), sigMat = myCov, bounds=myBounds)
```

calcb2 *calc_b2.R*

Description

Calculate lower bound or upper bound on power when considering only the two largest test statistic in magnitude, i.e. only |Z|_(J) and |Z|_(J-1).

Usage

```
calcb2(lower = TRUE, upper = FALSE, muVec, sigMat, bounds)
```

Arguments

lower	Boolean, whether to calculate lower bound.
upper	Boolean, whether to calculate upper bound.
muVec	Mean vector of test statistics under the alternative (assuming it's MVN).
sigMat	Covariance matrix of test statistics under the alternative (assuming it's MVN).
bounds	A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).

Value

A list with the elements:

allProbsLower	J*1 vector of all components summed to calculate lower bound.
lowerProb	Lower bound.
allProbsUpper	J*1 vector of all components summed to calculate upper bound.
upperProb	Upper bound.

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
myBounds <- set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
calcb2(muVec = c(1, 0, 0, 0, 0), sigMat = myCov, bounds=myBounds)
```

calc_exact_power *calc_exact_power.R*

Description

For detection boundary type tests, find the power given the rejection region bounds and specification of alternative. Do not use for sets larger than 5 elements, will be too slow.

Usage

```
calc_exact_power(bounds, sig_mat, muVec)
```

Arguments

bounds	A d=J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).
sig_mat	The covariance matrix of the test statistics under the alternative (assume multivariate normal).
muVec	The mean vector of the test statistics under the alternative (assume multivariate normal).

Value

A list with the elements:

power	Power under the given alternative.
errsum	Largest possible error from integration.
naSum	Number of NAs in calculating all integrals.
sumOverA	Matrix with power, errsum, naSum for each partition of the rejection region.

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
myBounds <- set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
calc_exact_power(bounds = myBounds, sig_mat = myCov, muVec = c(1, 0, 0, 0, 0))
```

createMj *Create the matrix that linearly transforms the vector of test statistics into a quantity amenable for pmvnorm.*

Description

Create the matrix that linearly transforms the vector of test statistics into a quantity amenable for pmvnorm.

Usage

```
createMj(j, size)
```

Arguments

j	The element of the vector that is the largest.
size	The length of the set.

Value

The transformation matrix of dimension (2J-1)*(2J-1)

Examples

```
createMj(j=3, size=5)
```

createMjk *Create the matrix that linearly transforms the vector of test statistics into a quantity amenable for pmvnorm.*

Description

Create the matrix that linearly transforms the vector of test statistics into a quantity amenable for pmvnorm.

Usage

```
createMjk(j, k, size)
```

Arguments

j	The element of the vector that is the largest.
k	The element of the vector that is the second largest.
size	The length of the set.

Value

The transformation matrix of dimension (2J-1)*(2J-1)

Examples

```
createMjk(j=3, k=4, size=5)
```

`performIntegralLower1` *Apply this function over 1:J to calculate each portion of the integral we need for the lower bound.*

Description

Apply this function over 1:J to calculate each portion of the integral we need for the lower bound.

Usage

```
performIntegralLower1(j, muVec, sigMat, lBounds1, uBounds1, lBounds2, uBounds2)
```

Arguments

`j`	Apply over this integer, the element that will be the largest in magnitude.
`muVec`	Mean vector of test statistics under the alternative (assuming it's MVN).
`sigMat`	Covariance matrix of test statistics under the alternative (assuming it's MVN).
`lBounds1`	A 2J-1 vector of lower bounds for the first integral (see paper).
`uBounds1`	A 2J-1 vector of upper bounds for the second integral (see paper).
`lBounds2`	A 2J-1 vector of lower bounds for the first integral (see paper).
`uBounds2`	A 2J-1 vector of upper bounds for the second integral (see paper).

Value

The value of the integration.

`performIntegralLower2` *Apply this function over all m, j not equal (order matters) to calculate each portion of the integral we need for the lower bound for calc_b2.*

Description

Apply this function over all m, j not equal (order matters) to calculate each portion of the integral we need for the lower bound for calc_b2.

Usage

```
performIntegralLower2(
  x,
  muVec,
  sigMat,
  lBounds1,
  uBounds1,
  lBounds2,
  uBounds2,
  lBounds3,
  uBounds3,
  lBounds4,
  uBounds4
)
```

Arguments

`x`	Apply over this 2*1 vector, the element that will be the largest in magnitude.
`muVec`	Mean vector of test statistics under the alternative (assuming it's MVN).
`sigMat`	Covariance matrix of test statistics under the alternative (assuming it's MVN).
`lBounds1`	A 2J-1 vector of lower bounds for the first integral (see paper).
`uBounds1`	A 2J-1 vector of upper bounds for the second integral (see paper).
`lBounds2`	A 2J-1 vector of lower bounds for the first integral (see paper).
`uBounds2`	A 2J-1 vector of upper bounds for the second integral (see paper).
`lBounds3`	A 2J-1 vector of lower bounds for the third integral (see paper).
`uBounds3`	A 2J-1 vector of upper bounds for the third integral (see paper).
`lBounds4`	A 2J-1 vector of lower bounds for the fourth integral (see paper).
`uBounds4`	A 2J-1 vector of upper bounds for the fourth integral (see paper).

Value

The value of the integration.

`performIntegralUpper1` *Apply this function over 1:J to calculate each portion of the integral we need for the upper bound.*

Description

Apply this function over 1:J to calculate each portion of the integral we need for the upper bound.

Usage

```
performIntegralUpper1(j, muVec, sigMat, lBounds1, uBounds1, lBounds2, uBounds2)
```

Arguments

`j`	Apply over this integer, the element that will be the largest in magnitude.
`muVec`	Mean vector of test statistics under the alternative (assuming it's MVN).
`sigMat`	Covariance matrix of test statistics under the alternative (assuming it's MVN).
`lBounds1`	A 3J-2 vector of lower bounds for the first integral (see paper), bounds will be longer than for performIntegralLower1.
`uBounds1`	A 3J-2 vector of upper bounds for the second integral (see paper).
`lBounds2`	A 3J-2 vector of lower bounds for the first integral (see paper).
`uBounds2`	A 3J-2 vector of upper bounds for the second integral (see paper).

Value

The value of the integration.

`performIntegralUpper2` *Apply this function over all m, j not equal (order matters) to calculate each portion of the integral we need for the lower bound for calc_b2.*

Description

Apply this function over all m, j not equal (order matters) to calculate each portion of the integral we need for the lower bound for calc_b2.

Usage

```
performIntegralUpper2(
  x,
  muVec,
  sigMat,
  lBounds1,
  uBounds1,
  lBounds2,
  uBounds2,
  lBounds3,
  uBounds3,
  lBounds4,
  uBounds4
)
```

Arguments

x	Apply over this 2*1 vector, the elements that will be the largest and second largest in magnitude.
muVec	Mean vector of test statistics under the alternative (assuming it's MVN).
sigMat	Covariance matrix of test statistics under the alternative (assuming it's MVN).
lBounds1	A 3J-2 vector of lower bounds for the first integral (see paper).
uBounds1	A 3J-2 vector of upper bounds for the second integral (see paper).
lBounds2	A 3J-2 vector of lower bounds for the first integral (see paper).
uBounds2	A J3J-2 vector of upper bounds for the second integral (see paper).
lBounds3	A 3J-2 vector of lower bounds for the third integral (see paper).
uBounds3	A 3J-2 vector of upper bounds for the third integral (see paper).
lBounds4	A 3J-2 vector of lower bounds for the fourth integral (see paper).
uBounds4	A 3J-2 vector of upper bounds for the fourth integral (see paper).

Value

The value of the integration.

---

set_BJ_bounds *set_BJ_bounds.R*

Description

Finds the boundary points of the rejection region for the BJ statistic when all elements in a set are independent.

Usage

```
set_BJ_bounds(alpha, J)
```

Arguments

alpha	Type I error of test.
J	Number of elements in set.

Value

A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).

Examples

```
set_BJ_bounds(alpha = 0.01, J=5)
```

---

set_GBJ_bounds *set_GBJ_bounds.R*

Description

Finds the boundary points of the rejection region for the GBJ statistic.

Usage

```
set_GBJ_bounds(alpha, J, sig_vec)
```

Arguments

alpha	Type I error of test.
J	Number of elements in set.
sig_vec	A vector generated from sigma[lower.tri(sigma)] where sigma is the correlation matrix of the test statistics.

Value

A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
```

sim_b1 *sim_R1.R*

Description

Simulate the probability of falling in the region used for the b1 lower bound or the b1 upper bound.

Usage

```
sim_b1(lower = TRUE, upper = TRUE, n, muVec, sigMat, bounds)
```

Arguments

lower	Boolean, if true sim lower bound.
upper	Boolean, if true sim upper bound.
n	Number of simulations.
muVec	Mean vector of test statistics under the alternative (assuming it's MVN).
sigMat	Covariance matrix of test statistics under the alternative (assuming it's MVN).
bounds	A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).

Value

A list with the elements:

lowerBound	Lower bound on power.
upperBound	Upper bound on power.

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
myBounds <- set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
sim_b1(n=5000, muVec = c(1, 0, 0, 0, 0), sigMat = myCov, bounds=myBounds)
```

sim_b2 *sim_b2.R*

Description

Simulate the probability of falling in the region used for the b2 lower bound or the b2 upper bound.

Usage

```
sim_b2(lower = TRUE, upper = FALSE, n, muVec, sigMat, bounds)
```

Arguments

lower	Boolean, if true sim lower bound.
upper	Boolean, if true sim upper bound.
n	Number of simulations.
muVec	Mean vector of test statistics under the alternative (assuming it's MVN).
sigMat	Covariance matrix of test statistics under the alternative (assuming it's MVN).
bounds	A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).

Value

A list with the elements:

lowerBound	Lower bound on power.
upperBound	Upper bound on power.

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
myBounds <- set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
sim_b2(n=5000, muVec = c(1, 0, 0, 0, 0), sigMat = myCov, bounds=myBounds)
```

sim_power_mvn *sim_power_mvn.R*

Description

Simulate power of detection boundary tests starting from multivariate normal test statistics.

Usage

```
sim_power_mvn(
  n,
  muVec,
  sigMat,
  nullSigMat = NULL,
  bounds = NULL,
  test = NULL,
  alpha
)
```

Arguments

n	Number of simulations.
muVec	Mean vector of test statistics under the alternative (assuming it's MVN).
sigMat	Covariance matrix of test statistics under the alternative (assuming it's MVN).
nullSigMat	Assumed correlation matrix of MVN under the null. Only need to specify if specifying test.
bounds	A J*1 vector of bounds on the magnitudes of the test statistics, where the first element is the bound for |Z|_(1) and the last element is the bound for |Z|_(J).
test	Either "GHC", "HC", "GBJ", or "BJ" or NULL. If provided, will calculate the p-value using the specified test and calculate power this way.
alpha	Level of the test.

Value

A list with the elements:

boundsPower	Power from using bounds approach.
testPower	Power from using specific test p-value approach.

Examples

```
myCov <- matrix(data=0.3, nrow=5, ncol=5)
diag(myCov) <- 1
myBounds <- set_GBJ_bounds(alpha = 0.01, J=5, sig_vec = myCov[lower.tri(myCov)])
sim_power_mvn(n=1000, muVec = c(1, 0, 0, 0, 0), sigMat = myCov, alpha=0.01)
```

sim_stats_mef *sim_power_indiv.R*

Description

Simulate power starting from individual-level data for multiple explanatory factor setting.

Usage

```
sim_stats_mef(
  B,
  sigSq,
  xMat,
  gMat,
  alphaVec,
  betaVec,
  decompTrue = NULL,
  checkpoint = FALSE
)
```

Arguments

B	Number of simulations.
sigSq	Variance of outcome.
xMat	Design matrix of non-genetic covariates, n*p.
gMat	Matrix of genotypes, n*J.
alphaVec	p*1 vector of regression coefficients for xMat.
betaVec	J*1 vector of regression coefficients for gMat.
decompTrue	The return value of a call to eigen() on the true covariance matrix. Can be null, in which case estimated covariance will be used.
checkpoint	Boolean, if true then print message every 50 simulations.

Value

A list with the elements:

zMat	B*J matrix of test statistics Z.
zVecGBJ	Check on Z statistics, vector should match first row of zMat.
iMat	Innovated statistics matrix also of dimension B*J.

Examples

```
xMat <- cbind(1, rnorm(n = 1000), rbinom(n = 1000, size=1, prob=0.5))
gMat <- matrix(data = rbinom(n=10000, size=2, prob=0.3), nrow=1000)
alphaVec <- c(1, 1, 1)
betaVec <- rep(0, 10)
sim_stats_mef(B=10000, sigSq = 1, xMat = xMat, gMat = gMat, alphaVec = alphaVec, betaVec = betaVec)
```

`sim_stats_mo` *Simulate power starting from individual-level data for multiple outcomes setting.*

Description

Simulate power starting from individual-level data for multiple outcomes setting.

Usage

```
sim_stats_mo(B, covY, xMat, gVec, alphaMat, gammaVec, checkpoint = FALSE)
```

Arguments

`B`	Number of simulations.
`covY`	Covariance matrix of outcomes.
`xMat`	Design matrix of non-genetic covariates, n*p.
`gVec`	n*1 vector of genotypes.
`alphaMat`	p*K vector of regression coefficients for xMat.
`gammaVec`	K*1 vector of regression coefficients for each outcome.
`checkpoint`	Boolean, if true then print message every 50 simulations.

Value

A list with the elements:

`zMat`	Matrix of test statistics Z.
`zVecGBJ`	Check on Z statistics, vector should match first row of zMat.
`iMat`	Innovated statistics using correlation matrix under the null.

Examples

```
## Not run:
covY <- matrix(data=0.3, nrow=10, ncol=10); diag(covY) <- 1
xMat <- cbind(1, rnorm(n = 1000), rbinom(n = 1000, size=1, prob=0.5))
gVec <- rbinom(n= 1000, size = 2, prob=0.3)
alphaMat <-matrix(data = 1, nrow=3, ncol=10)
gammaVec <- rep(0, 10)
sim_stats_mo(B=10000, covY = covY, xMat = xMat, gVec = gVec,
alphaMat = alphaMat, gammaVec = gammaVec)

## End(Not run)
```

Index