Package 'PCS'

May 7, 2026

Type Package

Title Calculate the Probability of Correct Selection (PCS)

Version 1.3

Date 2022-06-09

Depends statmod, multtest

Author Jason Wilson

Maintainer Jason Wilson <jason.wilson@biola.edu>

Description Given k populations (can be in thousands), what is the probability that a given subset of size t contains the true top t populations? This package finds this probability and offers three tuning parameters (G, d, L) to relax the definition.

License GPL-3

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2022-06-20 17:10:05 UTC

Contents

PCS-package 2
G.H.Quad 4
PCS.boot.par 5
PdCSGt.bootstrap.NP2 9
PdofCSGt.bootstrap5 10
PdofCSt.T1or2 11
PofCSLt.bootstrap5 12
PofCSt 13
tindep 14

Index **16**

PCS-package *Probability of Correct Selection (PCS)*

Description

These functions calculate the probability of correct selection (PCS) with G-best, d-best, and L-best selection as described in Cui & Wilson (2008) and Cui, Zhao, & Wilson (2008). The specific parameters (G,d,L), distributional assumptions (normal, Student's t, non-parametric), and calculation method (exact, bootstrap) are user-settable.

Details

Package:	PCS
Type:	Package
Version:	1.0
Date:	2009-04-15
License:	What license is it under?
LazyLoad:	yes

Probability of Correct Selection (PCS) is the probability of selecting the best t of k populations. This package allows the user to calculate the PCS for a given dataset. When k is large, even if the best t populations are significantly different from the rest, the PCS may be small due to sample variance. To address this issue, Cui & Wilson (2008, 2009) developed three tuning parameters whereby the definition of correct selection is modified (d-best, G-best, L-best) to more realistic and acceptable standards for large k problems. This package is the implementation of these three definitions, using different calculation methods.

The PCS package consists of three primary functions for users: PCS.boot.par, PCS.boot.np, and PCS.exact. PCS.boot.par and PCS.boot.np use parametric and non-parametric bootstraps, respectively, to calculate d-best, G-best, and L-best PCS. PCS.boot.par is the fastest function for large k problems. It is expected to be the most commonly used, as the parametric distributional (normal & Student's t) assumptions are reasonable and moderately robust (Cui & Wilson 2009). When k is large and the distributional assumptions are not met, then PCS.boot.np may be used. For information regarding the necessary sample size, see (Cui & Wilson 2009). When k is small to moderate, PCS.exact may be used to obtain PCS using the analytic formula.

Note

I would like to thank Xinping Cui for her support while creating this package, Thomas Girke for use of the UCR Bioinformatics Cluster, Bushi Wang for stress testing the code, and God for all the help on the way.

Author(s)

Jason Wilson, Maintainer: Jason Wilson <jason.wilson@biola.edu>

References

Cui, X. and Wilson, J. 2007. On How to Calculate the Probability of Correct Selection for Large k Populations. University of California, Riverside Statistics Department Technical Report 297. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjJm
Cui, X. and Wilson, J. 2008. On the Probability of Correct Selection for Large k Populations, with Application to Microarray Data. Biometrical Journal, 50:5, 870-883. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjE2YjI4MGRkNzA0ZTRjMDc
Cui, X. and Wilson, J. 2009. A Simulation Study on the Probability of Correct Selection for Large k Populations. Communications in Statistics - Simulation and Computation, 38:6. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjMxYTdjNjJkNzY3N
Cui, X.; Zhao, H. and Wilson, J. 2010. Optimized Ranking and Selection Methods for Feature Selection with Application in Microarray Experiments. Journal of Biopharmaceutical Statistics. Volume 20, No. 2, pp. 223-239. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjY0ZTJkNDNlMzNiNjE0ZDg

Examples

```
##### Small example for PCS.boot.par & PCS.exact
theta = c(4.2, 2.5, 2.3, 1.7, 1.5, 1.0)
theta = sort(theta)
T = c(1,2,3,5); G = 1:3; D = c(0.5,1,1.5); L = 1:2
PCS.boot.par(theta, T, G, D, L, B=100, SDE=1, dist="normal", df=14, trunc=6)
PCS.exact(theta, t=2, g=1, d=NULL, m=20, tol=1e-8)

##### Small example for PCS.boot.np
k=20 #number of populations
n=10 #sample size
SD=.1 #standard deviation
theta = seq(0,6,length.out=k)
X1 = rnorm(k*n,0,SD)   #Sample 1
X1 = matrix(X1,nrow=k,ncol=n,byrow=FALSE)
X2 = rnorm(k*n,theta,SD) #Sample 2, shifted
X2 = matrix(X2,nrow=k,ncol=n,byrow=FALSE)
T = c(1,2,3,5); G = 1:3; D = c(0.5,1,1.5)
PCS.boot.np(X1, X2, T, G, D, B=100, trunc=6)

##### Microarray example of t-statistics with PCS.boot.par
require(multtest)
data(golub)   #Load microarray data
sub = 500 #Subset index for speed
ans = tindep(golub[1:sub,1:27], golub[1:sub,28:38], flag=1) #Obtain t-statistics
golub.T = sort(abs(ans[,1]))   #Massage t-statistics
T=c(1,5,10,25,50,92); G=c(1,10,25,50,150); D=c(0,1,2) #Set PCS parameters
df=18    #Degrees of freedom from Satterthwaite approximation
sde=sqrt((18/(18-2))/19)  #Estimate SDE by MOM SD, divided by mean sample size
PCS.boot.par(golub.T, T, G, D, L=NULL, B=100, SDE=sde, dist="t", df=18) #Small B for speed

##### Microarray example of Golub's correlation statistics
##### (see reference) with PCS.boot.par
require(multtest)
data(golub)   #Load microarray data
```

```
Pgc <- function(x,y) {   #Function to calculate Golub's correlation statistics
  xbar1 = apply(x,1,mean)
  xbar2 = apply(y,1,mean)
  sd1   = apply(x,1,sd)
  sd2   = apply(y,1,sd)
  Pgc = abs((xbar1-xbar2))/(sd1+sd2)
  return(Pgc)
} #end function
sub = 500 #Subset index for speed
Pgc.gol = Pgc(golub[1:sub,1:27],golub[1:sub,28:38]) #Calculate correlation statistics
T=c(1,5,10,25,50,92); G=c(1,10,25,50,150); D=c(0,1,2) #Set PCS parameters
sde=0.20  #Obtained by simulation on Golub data
PCS.boot.par(Pgc.gol, T, G, D, L=NULL, B=100, SDE=0.2, dist="t", df=18) #Small B for speed

##### Microarray example using non-parametric bootstrap
require(multtest)
data(golub)   #Load microarray data
T=c(1,5,10); G=c(1,3,5); D=c(0,1,2)  #Set PCS parameters
sub = 100 #Subset index for speed
PCS.boot.np(golub[1:sub,1:27], golub[1:sub,28:38], T, G, D, B=10, trunc=6) #Small B for speed
```

`G.H.Quad` *Gauss-Hermite Quadrature function*

Description

Performs Gauss-Hermite Quadrature for an arbitrary number of nodes. Function for use in PofCSt.

Usage

```
G.H.Quad(x, m)
```

Arguments

`x`	is an m-vector upon which Gauss-Hermite Quadrature is to be applied
`m`	is the number of nodes desired for the Gauss-Hermite Quadrature

Details

This function uses gauss.quad.prob in the statmod package to perform Gauss-Hermite Quadrature inside of the PofCSt function.

Value

The result of the Gauss-Hermite Quadrature.

Author(s)

Jason Wilson <jason.wilson@biola.edu>

References

Hildebrand, F.B. 1956. Introduction to Numerical Analysis. McGraw-Hill.

See Also

PofCSt

PCS.boot.par *Probability of Correct Selection (PCS) Calculator*

Description

These functions calculate the probability of correct selection (PCS) with G-best, d-best, and L-best selection as described in Cui & Wilson (2008) and Cui, Zhao, & Wilson (2008). The specific parameters (G,d,L), distributional assumptions (normal, Student's t, non-parametric), and calculation method (exact, bootstrap) are user-settable.

Usage

```
PCS.boot.par(theta, T = 1:1, G = NULL, D = NULL, L = NULL, B = 100, SDE = 1,
dist = c("normal", "t"), df = 14, trunc = 6)
PCS.boot.np(X1, X2, T = 1, G = 1, D = NULL, B, trunc = 6)
PCS.exact(theta, t = 1, g = NULL, d = NULL, m = 20, tol = 1e-08)
```

Arguments

theta	Vector of statistics (or parameters) from which it is desired to select the top t of them
T	Vector of the number of statistics (or parameters) desired to be selected
t	The number of statistics (or parameters) desired to be selected (scalar)
G	Vector of G-best selection parameters
g	The G-best selection parameter (scalar)
D	Vector of d-best selection parameters
d	The d-best selection parameter (scalar)
L	Vector of L-best selection parameters
X1	k by n1 matrix of data. k is the number of populations and n1 the sample size of the first treatment.
X2	k by n2 matrix of data. k is the number of populations and n2 the sample size of the second treatment.
B	Bootstrap sample size
SDE	Standard error of the statistics theta (row-wise)
dist	Distributional assumption used for estimating PCS

df	Common degrees of freedom for one of the t-statistics in theta; the parameter is only used if dist="t"
trunc	Number of standard errors below the minimum selected population to disregard in the estimation of PCS; it is a truncation parameter to decrease run time
m	Number of nodes employed in the Gauss-Hermite quadrature
tol	Tolerance parameter to set the cut-off level for the inclusion of additional probability components in PCS

Details

The probability of correct selection (PCS) is a single statistic calculated on a vector of sample statistics whose interpretation is the probability that the top t populations selected according to the sample rank would be the true top t populations. PCS may be used in addition to, or instead of, multiple comparison procedures. The PCS package is an implementation of the formulas and ideas in Cui & Wilson (2008) and Cui, Zhao, & Wilson (2008).

After a multi-population experiment is conducted, whether with one, two, or more treatments, a vector of one sample statistic for each population is obtained. This vector is theta. Suppose theta has length k. The user will want to know the probability of correctly selecting some number t of the k populations. T is a vector of all such t desired to be checked. When k is large, the resulting probabilities can be uncomfortably small, even if the corresponding multiple-testing p-values are significant. In order to increase the resulting probabilities (PCS), three tuning parameters have been devised.

g is the number of populations which will be selected in addition to t. Thus, g+t populations are selected, only t of which must be the true top t populations in order to be considered correct. (Cui & Wilson 2007). G is a vector of such g's. Scalar g is used in the exact function (PCS.exact) and vector G is used in the bootstraps (PCS.boot.par & PCS.boot.np).

d does not directly specify a number of additional populations. Rather, d is the number of standard errors above and below the statistic whose rank is t (hereafter the "tth" statistic). The other statistics which fall in this range are considered close enough to the top t statistics which will be considered comparable to the tth population and therefore considered correct (since the tth statistic is an estimate of the tth population parameter). Thus, t populations are selected, but a random subset of the populations determined by d are considered "close enough" to the top t and are also considered correct (Cui & Wilson 2007). D is a vector of such d's. Scalar d is used in the exact function (PCS.exact) and vector D is used in the bootstraps (PCS.boot.par & PCS.boot.np). The above interpretation holds when the statistics of theta are standardized (SDE=1); when they are not, then d represents the number of units from the tth statistic, where the units are given by the data. Simulation studies suggest d>2 often produces significantly biased PCS when standardized statistics are used. Therefore, it is recommended that d<=2 be used when theta is standardized and d not greater than two standard errors when theta is not standardized.

l is the number of populations, out of the top t, which need to be included in the selected t to be considered a correct selection. Thus, t populations are selected, only l of which must be in the true top t, in order to be considered correct. (Cui & Wilson & Zahn 2008). L is a vector of such l. The parameter has not been implemented in the exact function (PCS.exact) or the non-parametric bootstrap (PCS.boot.np), only the parametric bootstrap (PCS.boot.par).

The run time of the function is directly proportional to the bootstrap sample size, B, (as well as the number of elements of T, D, G, and L). In practice, it is recommended that preliminary runs us a small B (e.g. B=100), in order to obtain estimates of PCS (off by several percent). Then, when the

desired parameters for final calculations are known, B should be run at B=10000, or more, depending on the application.
The SDE needs to be supplied by the user (this potential inconvenience is balanced by serving to make the code more general). If the statistics in theta are already standardized, as they often will be, then SDE=1. If the statistics in theta are not standardized, then a form of pooled standard error could be used for SDE. As for distributional assumptions, PCS.exact assumes a normal distribution on the elements of theta. PCS.boot.par allows for a "normal" or a "t" (Student's t, with degrees of freedom = df) distribution. PCS.boot.np is non-parametric, and requires significantly larger sample sizes in order to accurately estimate PCS (see Cui & Wilson 2009). In order to reduce run time, PCS.exact implements a tolerance parameter, tol, at four different levels of the calculations. It is defaulted conservatively to 1e-8 in order to produce answers exact to at least five decimal places of accuracy. PCS.boot.par and PCS.boot.np both implement a truncation parameter that reduces the size of the bootstrapped theta vector. It works by cutting the vector 'trunc' standard errors below the maximum Tth element of theta. trunc=6 produced no loss of accuracy on the runs attempted, while a loss of over 1 and m is the number of nodes employed. It is defaulted to m=20, which produced the same accuracy, to four or more decimal places, as m=1000.
As is customary in the Ranking and Selection literature, the three functions PCS.exact, PCS.boot.par, and PCS.boot.np all assume that the best statistics in theta are the largest. If the best statistics are the smallest, then a simple conversion will allow the functions to work equally well on the data. See the examples.

Value

For PCS.exact, the value is a scalar of the PCS for the parameters entered. For PCS.boot.par and PCS.boot.np, a matrix is returned whose rows are the entries of G, D, or L and whose columns are the entries of T. If more than one of G, D, or L is entered, then a list is returned, where the G element of the list is the G-best matrix, the d element is the d-best matrix, and the L element is the L-best matrix.

Author(s)

Jason Wilson, Maintainer: Jason Wilson <jason.wilson@biola.edu>

References

Cui, X. and Wilson, J. 2007. On How to Calculate the Probability of Correct Selection for Large k Populations. University of California, Riverside Statistics Department Technical Report 297. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjJm
Cui, X. and Wilson, J. 2008. On the Probability of Correct Selection for Large k Populations, with Application to Microarray Data. Biometrical Journal, 50:5, 870-883. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjE2YjI4MGRkNzA0ZTRjMDc
Cui, X. and Wilson, J. 2009. A Simulation Study on the Probability of Correct Selection for Large k Populations. Communications in Statistics: Simulation and Computation. 38:6. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjMxYTdjNjJkNzY3N
Cui, X.; Zhao, H. and Wilson, J. 2010. Optimized Ranking and Selection Methods for Feature Selection with Application in Microarray Experiments. Journal of Biopharmaceutical Statistics. Volume 20, No. 2, pp. 223-239. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjY0ZTJkNDNlMzNiNjE0ZDg

Golub, T. et al. 1999. Molecular Classification of Cancer Class Discovery and Class Prediction by Gene Expression Monitoring. Science. 286:531-537.

Examples

```
##### Small example for PCS.boot.par & PCS.exact
theta = c(1.0, 1.5, 1.7, 2.3, 2.5, 4.2)
T = c(1,2,3,5); G = 0:3; D = c(0.5,1,1.5); L = 1:2
PCS.boot.par(theta, T, G, D, L, B=100, SDE=1, dist="normal", trunc=6)
PCS.exact(theta, t=2, g=1, d=NULL, m=20, tol=1e-8)

#The above assumes the largest thetas are best.
#To select the smallest thetas, reverse the order of the indices, e.g.
theta.new = max(theta) - theta + min(theta)
T = c(1,2,3,5); G = 0:3; D = c(0.5,1,1.5); L = 1:2
PCS.boot.par(theta.new, T, G, D, L, B=100, SDE=1, dist="normal", trunc=6)
PCS.exact(theta.new, t=2, g=1, d=NULL, m=20, tol=1e-8)

##### Small example for PCS.boot.np
k=20 #number of populations
n=10 #sample size
SD=.1 #standard deviation
theta = seq(0,6,length.out=k)
X1 = rnorm(k*n,0,SD)   #Sample 1
X1 = matrix(X1,nrow=k,ncol=n,byrow=FALSE)
X2 = rnorm(k*n,theta,SD) #Sample 2, shifted
X2 = matrix(X2,nrow=k,ncol=n,byrow=FALSE)
T = c(1,2,3,5); G = 0:3; D = c(0.5,1,1.5); L = 1:2
PCS.boot.np(X1, X2, T, G, D, B=10, trunc=6)

##### Microarray example of t-statistics with PCS.boot.par
require(multtest)
data(golub)   #Load microarray data
sub = 500 #Subset index for speed
ans = tindep(golub[1:sub,1:27], golub[1:sub,28:38], flag=1) #Obtain t-statistics
golub.T = sort(abs(ans[,1]))   #Massage t-statistics
T=c(1,5,10,25,50,92); G=c(1,10,25,50,150); D=c(0,1,2) #Set PCS parameters
df=18    #Degrees of freedom from Satterthwaite approximation
sde=sqrt((18/(18-2))/19)  #Estimate SDE by MOM SD, divided by mean sample size
PCS.boot.par(golub.T, T, G, D, L=NULL, B=100, SDE=sde, dist="t", df=18) #Small B for speed

##### Microarray example of Golub's correlation statistics
##### (see reference) with PCS.boot.par
require(multtest)
data(golub)   #Load microarray data
Pgc <- function(x,y) {   #Function to calculate Golub's correlation statistics
  xbar1 = apply(x,1,mean)
  xbar2 = apply(y,1,mean)
  sd1   = apply(x,1,sd)
  sd2   = apply(y,1,sd)
  Pgc = abs((xbar1-xbar2))/(sd1+sd2)
  return(Pgc)
} #end function
```

```
sub = 500 #Subset index for speed
Pgc.gol = Pgc(golub[1:sub,1:27],golub[1:sub,28:38]) #Calculate correlation statistics
T=c(1,5,10,25,50,92); G=c(1,10,25,50,150); D=c(0,1,2) #Set PCS parameters
sde=0.20  #Obtained by simulation on Golub data
PCS.boot.par(Pgc.gol, T, G, D, L=NULL, B=100, SDE=0.2, dist="t", df=18) #Small B for speed

##### Microarray example using non-parametric bootstrap
require(multtest)
data(golub)   #Load microarray data
T=c(1,5,10); G=c(1,3,5); D=c(0,1,2)  #Set PCS parameters
sub = 100 #Subset index for speed
PCS.boot.np(golub[1:sub,1:27], golub[1:sub,28:38], T, G, D, B=10, trunc=6) #Small B for speed
```

`PdCSGt.bootstrap.NP2` *Non-Parametric Bootstrap for Computing G-best and d-best PCS*

Description

Non-parametric bootstrap for computing G-best and d-best PCS. This function is called by the wrapper PCS.boot.np.

Usage

```
PdCSGt.bootstrap.NP2(X1, X2, T, D, G, N, trunc = 6)
```

Arguments

`X1`	k by n1 matrix of data. k is the number of populations and n1 the sample size of the first treatment.
`X2`	k by n2 matrix of data. k is the number of populations and n2 the sample size of the second treatment.
`T`	Vector of the number of statistics (or parameters) desired to be selected
`D`	Vector of d-best selection parameters
`G`	Vector of G-best selection parameters
`N`	The bootstrap sample size
`trunc`	Number of standard errors below the minimum selected population to disregard in the estimation of PCS; it is a truncation parameter to decrease run time

Value

A matrix whose rows are the entries of G or D and whose columns are the entries of T. If both G and D are entered, then a list is returned, where the $G element is the G-best matrix, the $d element is the d-best matrix.

Author(s)

Jason Wilson <jason.wilson@biola.edu>

References

Cui, X. and Wilson, J. 2009. A Simulation Study on the Probability of Correct Selection for Large k Populations. Communications in Statistics: Simulation and Computation. 38:6. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjMxYTdjNjkNzY3N

See Also

PCS.boot.np

`PdofCSGt.bootstrap5` *Parametric Bootstrap for Computing G-best and d-best PCS*

Description

Parametric bootstrap for computing G-best and d-best PCS. This function is called by the wrapper PCS.boot.par.

Usage

```
PdofCSGt.bootstrap5(theta, T, D, G, B, SDE, dist = c("normal", "t"),
df = 14, trunc = 6, est.names = c("O"))
```

Arguments

`theta`	Vector of statistics (or parameters) from which it is desired to select the top t of them
`T`	Vector of the number of statistics (or parameters) desired to be selected
`D`	Vector of d-best selection parameters
`G`	Vector of G-best selection parameters
`B`	Bootstrap sample size
`SDE`	Standard error of the statistics theta (row-wise)
`dist`	Distributional assumption used for estimating PCS
`df`	Common degrees of freedom for one of the t-statistics in theta; the parameter is only used if dist="t"
`trunc`	Number of standard errors below the minimum selected population to disregard in the estimation of PCS; it is a truncation parameter to decrease run time
`est.names`	Kind of shrinkage estimator employed. Default estimator is "O" for the Olkin estimator. Other estimators will be considered for future releases.

Value

An array, the non-empty part of which is a matrix whose rows are the entries of G or D and whose columns are the entries of T. If both G and D are entered, then a list is returned, where the $G element is the G-best matrix, the $d element is the d-best matrix.

Author(s)

Jason Wilson <jason.wilson@biola.edu>

References

Cui, X. and Wilson, J. 2007. On How to Calculate the Probability of Correct Selection for Large k Populations. University of California, Riverside Statistics Department Technical Report 297. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjJm

See Also

PCS.boot.par

PdofCSt.T1or2 *Exact PCS, When Selecting t=1 or t=2 or More Populations*

Description

PdofCSt.T1or2 calls PofCSt to implement the d-best PCS formula for the case of t=1 or t=2 populations. It is called by PdofCSt.cyc2, which implements the d-best PCS formula for general t populations. PofCSGt calls PofCSt to implement the G-best formula for general t. These functions are modular and implement the time reduction techniques of Cui and Wilson (2007).

Usage

```
PdofCSt.T1or2(theta, T, d, m = 20, tol = 1e-08)
PdofCSt.cyc2(theta, T, d, m = 20, tol = 1e-08)
PofCSGt(theta, T, Gd, m = 20, tol = 1e-08)
```

Arguments

theta	Vector of statistics (or parameters) from which it is desired to select the top t of them
T	The number of statistics (or parameters) desired to be selected
d	The d-best selection parameter (scalar)
Gd	The G-best selection parameter (scalar)
m	Number of nodes employed in the Gauss-Hermite quadrature
tol	Tolerance parameter to set the cut-off level for the inclusion of additional probability components in PCS

Value

The probability of correct selection.

Author(s)

Jason Wilson <jason.wilson@biola.edu>

References

Cui, X. and Wilson, J. 2007. On How to Calculate the Probability of Correct Selection for Large k Populations. University of California, Riverside Statistics Department Technical Report 297. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjJm

See Also

PofCSt, PdofCSt.T1or2, PdofCSt.cyc2, PofCSGt

`PofCSLt.bootstrap5` *Parametric Bootstrap for Computing L-best PCS*

Description

Parametric bootstrap for computing L-best PCS. This function is called by the wrapper PCS.boot.np.

Usage

```
PofCSLt.bootstrap5(theta, T, L, B, SDE, dist = c("normal", "t"),
     df = 14, trunc = 6, est.names = c("O"))
```

Arguments

`theta`	Vector of statistics (or parameters) from which it is desired to select the top t of them
`T`	Vector of the number of statistics (or parameters) desired to be selected
`L`	Vector of L-best selection parameters
`B`	Bootstrap sample size
`SDE`	Standard error of the statistics theta (row-wise)
`dist`	Distributional assumption used for estimating PCS
`df`	Common degrees of freedom for one of the t-statistics in theta; the parameter is only used if dist="t"
`trunc`	Number of standard errors below the minimum selected population to disregard in the estimation of PCS; it is a truncation parameter to decrease run time
`est.names`	Kind of shrinkage estimator employed. Default estimator is "O" for the Olkin estimator. Other estimators will be considered for future releases.

Value

An array, the non-empty part of which is a matrix whose rows are the entries of L and whose columns are the entries of T.

Author(s)

Jason Wilson <jason.wilson@biola.edu>

References

Cui, X.; Zhao, H. and Wilson, J. 2010. Optimized Ranking and Selection Methods for Feature Selection with Application in Microarray Experiments. Journal of Biopharmaceutical Statistics. Volume 20, No. 2, pp. 223-239. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjY0ZTJkNDNlMzNiNjE0ZDg

See Also

PCS.boot.par

`PofCSt` *Probability of Correct Selection (PCS) for Selecting t out of k Populations*

Description

Implementation of the Gupta & Liang (1998) formula for computing the probability of correct selection (PCS) for selecting t out of k populations. The results are exact up to a user-settable tolerance parameter. This function is modular and is called by PdofCSt.T1or2, PdofCSt.cyc2, and PofCSGt.

Usage

```
PofCSt(theta, T, m, tol = 1e-07)
```

Arguments

`theta`	Vector of statistics (or parameters) from which it is desired to select the top t of them
`T`	The number of statistics (or parameters) desired to be selected
`m`	Number of nodes employed in the Gauss-Hermite quadrature
`tol`	Tolerance parameter to set the cut-off level for the inclusion of additional probability components in PCS

Details

The analytic formula for computing PCS for t of k populations is an integral whose integrad is the product of normal densities. This function obtains the appropriate values and computes the integral using a Gauss-Hermite quadrature. See equation 2.4 of Gupta (1998).

Value

The probability of correct selection.

Author(s)

Jason Wilson <jason.wilson@biola.edu>

References

Cui, X. and Wilson, J. 2007. On How to Calculate the Probability of Correct Selection for Large k Populations. University of California, Riverside Statistics Department Technical Report 297. https://docs.google.com/a/biola.edu/viewer?a=v&pid=sites&srcid=YmlvbGEuZWR1fGphc29ud2lsc29ufGd4OjJm
Gupta, S.S. and Liang, T.C. 1998. Simultaneous lower confidence bounds for probabilities of correct selections. Journal of Statistical Planning and Inference. 72(1-2), 279-290.

See Also

PdofCSt.T1or2, PdofCSt.cyc2, PofCSGt

`tindep` *Standardized Scores for Two Independent Samples*

Description

Calculate the independent two-sample Welch t-statistics for k populations simultaneously. This function is used in PCS.bootstrap.np. It may also be used to summarize data from two sample experiments for use in PCS.exact and PCS.bootstrap.par.

Usage

```
tindep(X, Y, flag = 0)
```

Arguments

`X`	k by n1 matrix of data. k is the number of populations and n1 the sample size of the first treatment.
`Y`	k by n2 matrix of data. k is the number of populations and n2 the sample size of the second treatment.
`flag`	Determines whether the output is t-statistics (flag=0) or t-statistics with p-values (flag!=0)

Details

X & Y are the data matrices for input. When called inside of the function PCS.bootstrap.np, or when used to obtain t-statistics for use with PCS.exact and PCS.bootstrap.par, setting 'flag'=0 will return the needed vector of t-statistics. If flag!=0, then the function 'mt.rawp2adjp' from the multtest library is called, producing multiple comparison adjustd pvalues ("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD", "BH", "BY"). The t-statistics and p-values are stored in a matrix.

Value

If flag = 0, the result is a k by 1 vector of t-statistics. If flag != 0, the result is a k by 7 vector of t-statistics and pvalues (see details).

Author(s)

Jason Wilson <jason.wilson@biola.edu>

See Also

PCS.boot.np

Examples

```
### Small example
k=20 #number of populations
n=10 #sample size
SD=.1 #standard deviation
theta = seq(0,6,length.out=k)
X1 = rnorm(k*n,0,SD)   #Sample 1
X1 = matrix(X1,nrow=k,ncol=n,byrow=FALSE)
X2 = rnorm(k*n,theta,SD) #Sample 2, shifted
X2 = matrix(X2,nrow=k,ncol=n,byrow=FALSE)
tindep(X1,X2, flag=1)

### Microarray example
#require(multtest)
data(golub)
sub = 500 #Subset index for speed
golub.Tp = tindep(golub[1:sub,1:27], golub[1:sub,28:38], flag=0) #Obtain t-statistics + p-values

ans = tindep(golub[1:sub,1:27], golub[1:sub,28:38], flag=1)  #Obtain t-statistics
golub.T = sort(abs(ans[,1]))
```

Index

∗ **htest**
PCS-package, 2
PCS.boot.par, 5
PdCSGt.bootstrap.NP2, 9
PdofCSGt.bootstrap5, 10
PdofCSt.T1or2, 11
PofCSLt.bootstrap5, 12
PofCSt, 13
tindep, 14
∗ **math**
G.H.Quad, 4
PCS-package, 2

G.H.Quad, 4

PCS *(*PCS-package*)*, 2
PCS-package, 2
PCS.boot.np, *10*, *15*
PCS.boot.np *(*PCS.boot.par*)*, 5
PCS.boot.par, 5, *11*, *13*
PCS.exact *(*PCS.boot.par*)*, 5
PdCSGt.bootstrap.NP2, 9
PdofCSGt.bootstrap5, 10
PdofCSt.cyc2, *12*, *14*
PdofCSt.cyc2 *(*PdofCSt.T1or2*)*, 11
PdofCSt.T1or2, 11, *12*, *14*
PofCSGt, *12*, *14*
PofCSGt *(*PdofCSt.T1or2*)*, 11
PofCSLt.bootstrap5, 12
PofCSt, *5*, *12*, 13

tindep, 14