Package 'popKorn'

February 20, 2015

Type Package

Title For interval estimation of mean of selected populations

Version 0.3-0

Date 2014-07-04

Author Vik Gopal, Claudio Fuentes

Maintainer Vik Gopal <stavg@nus.edu.sg>

Depends R (>= 3.0.0), boot

Suggests plotrix

Description Provides a suite of tools for various methods of estimating confidence intervals for the mean of selected populations.

License MIT + file LICENSE

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-11 16:45:25

R topics documented:

popKorn-package *Interval estimation for selected populations*

Description

Interval estimation for selected populations

Details

Package:	popKorn
Type:	Package
Version:	0.3-0
Date:	2014-07-04
License:	MIT
LazyLoad:	yes

This package provides routines for estimating the interval for the mean of selected populations.

Author(s)

Claudio Fuentes, Vik Gopal

asymmetricIntervals *Compute Asymmetric Intervals*

Description

This function will compute asymmetric intervals for the mean of the selected populations.

Usage

```
asymmetricIntervals(X, alpha = 0.05, k = 2, var = NULL, eps = 0.1)
```

Arguments

X	is a matrix or data frame that contains the responses. Each column represents a different population.
alpha	denotes the significance level of the intervals to be formed.
k	corresponds to the number of populations to be selected.
var	denotes the common variance of the populations from which the data is drawn. If this is NULL (the default), then the variance will be estimated from the data. If it is known, then it should be provided as a scalar.
eps	The grid size that is to be set up.

Details

This function will compute the optimal lambda and c to be used to shrink the interval for the selected populations.

Value

This function returns a matrix with k rows and 3 columns. This is similar to the output of the predict.lm function of R.

References

Claudio Fuentes, George Casella and Martin Wells (2013). Interval estimation for the mean of the selected populations (Submitted).

Vik Gopal and Claudio Fuentes (2013). kPop: An R package for interval estimation of selected populations. useR! 2013.

See Also

bonferroniIntervals, bootstrapIntervals

Examples

```
set.seed(18)
p <- 10; n <- 10
Xmat <- matrix(rnorm(p*n), nrow=n, ncol=p)
colnames(Xmat) <- paste("p.", 1:p, sep="")
asymmetricIntervals(Xmat, alpha=0.1, k=4)
```

---

bonferroniIntervals *Compute Bonferroni Intervals*

Description

This function will compute apply the Bonferroni correction to the selected populations.

Usage

```
bonferroniIntervals(X, alpha = 0.05, k = 2)
```

Arguments

X	is a matrix or data frame that contains the responses. Each column represents a different population.
alpha	denotes the significance level of the intervals to be formed.
k	corresponds to the number of populations to be selected.

Details

If there are p populations, then the Bonferroni correction will be applied by using α/p instead of just p.

Value

The function returns a matrix with k rows and 3 columns. This is similar to the output of the predict.lm function of R.

See Also

asymmetricIntervals, bootstrapIntervals

Examples

```
set.seed(18)
p <- 10; n <- 10
Xmat <- matrix(rnorm(p*n), nrow=n, ncol=p)
colnames(Xmat) <- paste("p.", 1:p, sep="")
bonferroniIntervals(Xmat, alpha=0.1, k=4)
```

---

bootstrapIntervals *Compute Bootstrap Intervals*

Description

This function will apply the Bonferroni correction to bootstrap intervals of the mean of the selected populations.

Usage

```
bootstrapIntervals(X, alpha = 0.05, k = 2, R = 10,
  return.obj = "intervals", type = "basic", ...)
```

Arguments

X	is a matrix or data frame that contains the responses. Each column represents a different population.
alpha	denotes the significance level of the intervals to be formed.
k	corresponds to the number of populations to be selected.
R	denotes the number of bootstrap replicate samples to produce.
return.obj	is a character vector of length 1, indicating if this function should return the output of the boot() function, or proceed to compute the bootstrap confidence intervals and return those.
type	is a character vector of length one. It should be one of the following: "norm","basic", "stud", "perc" or "bca".
...	denotes further arguments that will be passed to the boot function.

Details

The bootstrap that is carried out is the stratified bootstrap, since there are p population in consideration. Within each population, sampling with replacement is carried out, and the largest k sample means are returned.

The user can use any of the 5 confidence interval methods that are present in the boot.ci function. However, a Bonferroni correction will be carried out in order to ensure that the intervals hold simultaneously.

Value

If `return.obj` is set to be "boot", then the function returns an object of class "boot". Otherwise, if `return.obj` is set to be "intervals", then this function returns a matrix with k rows and 3 columns. This is similar to the output of the predict.lm function of R.

See Also

`bonferroniIntervals`, `asymmetricIntervals`

Examples

```
set.seed(18)
p <- 10; n <- 10
Xmat <- matrix(rnorm(p*n), nrow=n, ncol=p)
colnames(Xmat) <- paste("p.", 1:p, sep="")
bootstrapIntervals(Xmat, alpha=0.1, k=4)
```

---

eqnForMin *Evaluate the function to be minimised*

Description

This will evaluate the appropriate function for determining the c value in the confidence interval for $X_(1)$.

Usage

```
eqnForMin(c.val, lambda = 0.5, alpha = 0.05, min.loc = "infty", n, p,
  k = 1, var.known = TRUE)
```

Arguments

`c.val`	The value at which to evaluate the appropriate function.
`lambda`	The value of lambda under consideration. This must be a scalar between 0 and 1.
`alpha`	The desired confidence coefficient.
`min.loc`	The location of the minimum, either at 'zero' or 'infty'.

n	The number of replications per population.
p	The number of populations considered. This must be present if min.loc is equal to 'zero'.
k	The number of populations selected.
var.known	A logical flag indicating if the variance of the observations is known exactly. It is TRUE by default.

Details

This function will choose the correct equation to use for determining the smallest c-value that maintains the desired coverage probability. Note that this function does *not* do the minimization. That procedure is done by optimalC.

There are essentially 8 different cases to consider. They correspond to the cases when the variance is known or unknown, when the number of populations selected is greater than 1 or equal to 1, and when the minimum of the equation is located at infinity or 0.

Value

The function returns a scalar value.

---

exactCoverageProb *Evaluate exact coverage probability*

Description

This function will evaluate the exact coverage probability, as given in equation (4) on page 7 of the paper. See Details section.

Usage

```
exactCoverageProb(c.vec, theta.diff, lambda, c.val, sigma.2 = 1, n = 1)
```

Arguments

c.vec	This is a vector of length 2. It consists of the lower and upper limits in the integral. Checking is carried out to ensure that is of length two, and that 0 <= c.vec[1] <= c.vec[2]. This parameter is ignored if lambda is not missing.
theta.diff	A vector of length p-1, where p is the number of populations of treatments. Coordinate [i] in theta.diff corresponds to $\theta_i - \theta_{i+1}$. See genDelMat.
lambda	In case the user wishes to use the shrinkage version, this parameter should be specified. It must be between 0 and 1.
c.val	In case lambda is specified, this must not be missing. This will be combined with lambda to create a c.vec. This very function will then call itself.
sigma.2	The known variance of the error terms.
n	The number of replications per population.

Details

This function evaluates the coverage probability for an interval defined by $(X_{(1)} - c_2, X_{(1)} + c_1)$. Note that, as specified in the reference paper, we must have that $0 \leq c_1 \leq c_2$. This function will call integrate2. Please note the ordering of the elements in the `c.vec` argument: the first element corresponds to the upper limit of the interval, and to the negative of the lower limit of the integral.

Value

The function returns a scalar value that is the value of the exact coverage coverage probability defined in equation (4) of page 7.

See Also

integrate2, integrand

Examples

```
del1 <- c(2, 4)
exactCoverageProb(c(1.1,1.3), del1)
exactCoverageProb(theta.diff=c(2,3,4), lambda=0.9, c.val=2)
```

---

genDelMat *Generate delta matrix*

Description

This function will generate the matrix of deltas, as specified in the paper. See Details section.

Usage

```
genDelMat(theta.diff, sigma.2 = 1, n = 1)
```

Arguments

`theta.diff`	A vector of length p-1, where p is the number of populations of treatments. Coordinate [i] in theta.diff corresponds to $\theta_i - \theta_{i+1}$.
`sigma.2`	The known variance of the error terms.
`n`	The number of replications in each population.

Details

As specified in the paper, we can assume that the thetas are in a decreasing order, meaning that $\theta_1 \geq \theta_2, \ldots, \theta_n$. It follows that all the components of the theta.diff vector must be positive. Note that the delta matrix in the paper is a scaled version of the differences between the thetas.

Value

The function returns a matrix with p rows and p columns, that contains the $delta_{ij}$'s, as described in the paper.

See Also

exactCoverageProb, integrand

Examples

```
del1 <- c(2, 4)
genDelMat(del1)
```

---

integrand *Evaluate integrand alone*

Description

This function will evaluate the integrand in the expression for the exact coverage probability.

Usage

```
integrand(z, mat1)
```

Arguments

z	This is a real number between -Inf and Inf.
mat1	This is the matrix of the delta values. This matrix can be generated using the genDelMat function.

Value

The function returns a scalar value.

See Also

exactCoverageProb, integrate2

`integrate2` *Evaluate integral*

Description

This function will evaluate the integral in equation (4) on page 7 of the paper. See Details section.

Usage

```
integrate2(limit.vec = c(-3, 3), theta.diff, sigma.2 = 1, n = 1)
```

Arguments

`limit.vec`	This is a vector of length 2. It consists of the lower and upper limits in the integral. Checking is carried out to ensure that is of length two, and that limit.vec[1] <= limit.vec[2].
`theta.diff`	A vector of length p-1, where p is the number of populations of treatments. Coordinate [i] in theta.diff corresponds to $\theta_i - \theta_{i+1}$. See `genDelMat`.
`sigma.2`	The known variance of the error terms.
`n`	The number of replications per population.

Details

This function evaluates the integral, and works with the lower and upper limits that it is given. If one desires to compute the coverage probability for an interval defined by $X_{(1)} \pm c$, then the user should look at the function exactCoverageProb in this package.

Value

The function returns a scalar value that is the value of the integral in equation (4) of page 7, defined by the lower and upper limits provided here.

See Also

exactCoverageProb, integrand

Examples

```
del1 <- c(2, 4)
integrate2(c(-1.1,1.3), del1)
```

`optimalC` *Derive the optimal c*

Description

Derives the optimal c value for a given lambda.

Usage

```
optimalC(lambda, alpha = 0.05, min.loc = "infty", n, p, k = 1,
  var.known = TRUE)
```

Arguments

`lambda`	The value of lambda under consideration. This must be a vector with values between 0 and 1.
`alpha`	The desired confidence coefficient.
`min.loc`	The location of the minimum, either at 'zero' or 'infty'.
`n`	The number of replications per population.
`p`	The number of populations considered. This must be present if min.loc is equal to 'zero'.
`k`	The number of populations selected.
`var.known`	A logical flag indicating if the variance of the observations is known exactly. It is TRUE by default.

Details

This function will choose the correct equation to use for and use 'uniroot' to find the c-value that corresponds to the desired alpha-level.

Value

The function returns a vector of length equal to that of lambda.

See Also

`optimalLambda`

`optimalLambda` *Derive the optimal lambda*

Description

Derives the optimal lambda for a given alpha.

Usage

```
optimalLambda(alpha, n, p, k = 1, var.known = TRUE)
```

Arguments

`alpha`	The desired confidence coefficient.
`n`	The number of replications per population.
`p`	The number of populations considered. This must be present if min.loc is equal to 'zero'.
`k`	The number of populations selected.
`var.known`	A logical flag indicating if the variance of the observations is known exactly. It is TRUE by default.

Details

This will find the optimal lambda to be used for the shrinkage of confidence intervals.

Value

The function returns a scalar value.

See Also

`optimalC`

`optimalLambdaC` *Derive the optimal lambda and c value*

Description

Derives the optimal lambda and c-value for a given configuration.

Usage

```
optimalLambdaC(alpha = 0.05, n, p, k = 1, var.known = TRUE, eps = 0.1)
```

Arguments

alpha	The desired confidence coefficient.
n	The number of replications per population.
p	The number of populations considered. This must be present if min.loc is equal to 'zero'.
k	The number of populations selected.
var.known	A logical flag indicating if the variance of the observations is known exactly. It is TRUE by default.
eps	The grid size that is to be set up.

Details

This function will return the optimal lambda and c-value to be used, using a grid search.

There are essentially 2 different cases to consider. They correspond to the cases when the variance is known or unknown.

Value

The function returns a list with two components, lambda and c.val that are optimal.

Index