Package 'TEAM'

September 13, 2019

Type Package

Title Multiple Hypothesis Testing on an Aggregation Tree Method

Version 0.1.0

Author John Pura, Cliburn Chan, and Jichun Xie

Maintainer John Pura <john.pura@duke.edu>

Description An implementation of the TEAM algorithm to identify local differences between two (e.g. case and control) independent, univariate distributions, as described in J Pura, C Chan, and J Xie (2019) <arXiv:1906.07757>. The algorithm is based on embedding a multiple-testing procedure on a hierarchical structure to identify high-resolution differences between two distributions. The hierarchical structure is designed to identify strong, short-range differences at lower layers and weaker, but long-range differences at increasing layers. TEAM yields consistent layer-specific and overall false discovery rate control.

License GPL-2

Encoding UTF-8

LazyData true

Imports plyr,ggplot2,ks

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2019-09-13 10:10:02 UTC

R topics documented:

chunk.sum *Chunk.Sum function*

Description

Rolling Sum over distinct chunks

Usage

```
chunk.sum(v, n, na.rm = TRUE)
```

Arguments

v	Numeric Vector
n	Size of chunk
na.rm	Remove NAs (default=TRUE)

est.c.hat *Estimate threshold function*

Description

Estimate threshold to control FDR in multiple testing procedure

Usage

```
est.c.hat(l, n, theta0, x.l, c.hats, alpha, m.l)
```

Arguments

l	Layer
n	Number of pooled case and control observations in each layer 1 bin
theta0	Nominal boundary level for binomial parameter at layer 1
x.l	Vector of case counts in each bin
c.hats	Previous c.hats calculated from layers 1 to l-1
alpha	Nominal FDR level
m.l	Number of leaf hypotheses at layer l

est.FDR.hat.l *Calculate FDR*

Description

Step-down multiple-testing procedure

Usage

```
est.FDR.hat.l(min.x, max.x, c.prev, n.l, x.l, theta0, l)
```

Arguments

min.x	lower limit of searching range
max.x	upper limit of searching range
c.prev	Previous c.hat from layer l-1
n.l	Vector of number of pooled observations in layer l bins
x.l	Vector of case counts in each bin at layer l
theta0	Nominal boundary level for binomial parameter at layer l
l	Layer

expand.mat *Enumerate binomial support*

Description

Enumerate possible counts for calculating binomial probability

Usage

```
expand.mat(mat, vec)
```

Arguments

mat	Matrix
vec	Numeric Vector

splitNoOverlap *splitNoOverlap function*

Description

Split a vector into distinct chunks of specified size

Usage

```
splitNoOverlap(vec, seg.length)
```

Arguments

vec	Numeric Vector
seg.length	Number of distinct chunks to split vec

TEAM *Testing on an Aggregation Tree Method*

Description

This function performs multiple testing embedded in a hierarchical structure in order to identify local differences between two independent distributions (e.g. case and control).

Usage

```
TEAM(x1, x2, theta0 = length(x2)/length(c(x1, x2)), K = 14,
  alpha = 0.05, L = 3)
```

Arguments

x1	Numeric vector of N1 control observations
x2	Numeric vector of N2 case observations
theta0	Nominal boundary level for binomial parameter - default is N2/(N1+N2)
K	log2 number of bins
alpha	Nominal false discovery rate (FDR) level
L	Number of layers in the aggregation tree

Value

List containing the discoveries (S.list) in each layer and the estimated layer-specific thresholds (c.hats)

References

Pura J. Chan C. Xie J. Multiple Testing Embedded in an Aggregation Tree to Identify where Two Distributions Differ. https://arxiv.org/abs/1906.07757

Examples

```
set.seed(1)
# Simulate local shift difference for each population from mixture of normals
N1 <- N2 <- 1e6
require(ks) #loads rnorm.mixt function
#Controls
x1 <- rnorm.mixt(N1,mus=c(0.2,0.89),sigmas=c(0.04,0.01),props=c(0.97,0.03))
#Cases
x2 <- rnorm.mixt(N2,mus=c(0.2,0.88),sigmas=c(0.04,0.01),props=c(0.97,0.03))
res <- TEAM(x1,x2,K=14,alpha=0.05,L=3)
#Discoveries in each layer - Each element is an growing set of
#indices captured at each layer
res$S.list
#Map back final discoveries in layer 3 to corresponding regions
levels(res$dat$quant)[res$S.list[[3]]]
```

`valid.counts` *Valid counts*

Description

Enumerate matrix of valid counts for a vector of values

Usage

```
valid.counts(x, c.prev)
```

Arguments

`x`	Vector
`c.prev`	Calculated chat from layer l-1

Index