Package 'permDep'

July 5, 2018

Title Permutation Tests for General Dependent Truncation

Version 1.0.2

Description Implementations of permutation approach to hypothesis testing for quasi-independence of truncation time and failure time. The implemented approaches are powerful against non-monotone alternatives and thereby offer protection against erroneous assumptions of quasi-independence. The proposed tests use either a conditional or an unconditional method to evaluate the permutation p-value. The conditional method was first developed in Tsai (1980) <doi:10.2307/2336059> and Efron and Petrosian (1992) <doi:10.1086/171931>. The unconditional method provides a valid approximation to the conditional method, yet computationally simpler and does not hold fixed the size of each risk sets. Users also have an option to carry out the proposed permutation tests in a parallel computing fashion.

Depends R (>= 3.4.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

URL http://github.com/stc04003/permDep

BugReports http://github.com/stc04003/permDep/issues

Imports BB, survival, parallel

Author Sy Han (Steven) Chiou [aut, cre]

Maintainer Sy Han (Steven) Chiou <schiou@utdallas.edu>

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-07-05 05:50:02 UTC

R topics documented:

permDep-package *Permutation Test for General Dependent Truncation*

Description

Implementations of permutation approaches to test for quasi-independence. The implemented approaches are powerful against non-monotone alternatives and thereby offer protection against erroneous assumptions of quasi-independence. The proposed tests use either a conditional or a unconditional method to evaluate the permutation p-value. Users also have an option to carry out the proposed permutation tests in a parallel computing fashion.

Author(s)

Maintainer: Sy Han (Steven) Chiou <schiou@utdallas.edu>

References

Chiou, S.H., Qian, J., and Betensky, R.A. (2017). Permutation Test for General Dependent Truncation. *Techreport*

See Also

Useful links:

- http://github.com/stc04003/permDep
- Report bugs at http://github.com/stc04003/permDep/issues

permDep *Permutation test for general dependent truncation*

Description

Perform permutation test based on conditional or unconditional approach.

Usage

```
permDep(trun, obs, permSize, cens, sampling = c("conditional",
  "unconditional", "is.conditional", "is.unconditional"), kendallOnly = FALSE,
  minp1Only = FALSE, minp2Only = FALSE, nc = ceiling(detectCores()/2),
  seed = NULL)
```

Arguments

`trun`	is the left truncation time.
`obs`	is the observed failure time.
`permSize`	is the number of permutations.
`cens`	is the status indicator; 0 = censored, 1 = event.
`sampling`	a character string specifying the sampling method used in permutation. The following are permitted: `conditional` conditional permutation; `uconditional` unconditional permutation; `is.conditional` importance sampling version of conditional permutation; `is.uconditional` importance sampling version of unconditional permutation;
`kendallOnly, minp1Only, minp2Only`	optional values indicating which test statistics to be used. If all leave as `FALSE`, `permDep` will use all three test statistics in each permutation.
`nc`	is the number of cores used in permutation. When `nc > 1`, permutation is carried out with parallel computing.
`seed`	an optional vector containing random seeds to be used to generate permutation samples. Random seeds will be used when left unspecified.

Value

A list containing output with the following components:

obsKen the observed p-value using Kendall's tau test statistic.

obsP1 the observed p-value using minp1 test statistic.

obsP2 the observed p-value using minp2 test statistic.

obsTest1 the observed minp1 test statistic.

obsTest2 the observed minp2 test statistic.

permKen Kendall's tau test statistics from permutation samples.

permP1 minp1 test statistics from permutation samples.

permP2 minp2 test statistics from permutation samples.

References

Chiou, S.H., Qian, J., and Betensky, R.A. (2017). Permutation Test for General Dependent Truncation. *Tech-report*

Examples

```
simDat <- function(n) {
  k <- s <- 1
  tt <- xx <- yy <- cc <- delta <- rep(-1, n)
  while(k <= n){
    tt[k] <- runif(1, 0, 3.5)
```

```
    xx[k] <- 1.95 + 0.65 * (tt[k] - 1.25)^2 + rnorm(1, sd = 0.1)
    cc[k] <- runif(1, 0, 10)
    delta[k] <- (xx[k] <= cc[k])
    yy[k] <- pmin(xx[k], cc[k])
    s <- s + 1
    if(tt[k] <= yy[k]) k = k+1
  }
 data.frame(list(trun = tt, obs = yy, delta = delta))
}

set.seed(123)
dat <- simDat(50)
B <- 20

## Perform conditional permutation with Kendall's tau, minp1 and minp2
set.seed(123)
system.time(fit <- with(dat, permDep(trun, obs, B, delta, nc = 1)))
fit
```

Index