Package ‘permGS’

August 8, 2017

Title Permutational Group Sequential Test for Time-to-Event Data
Version 0.2.5
Date 2017-08-06
Maintainer Matthias Brueckner <matthias.brueckner@posteo.de>
Description Permutational group-sequential tests for time-to-event data based on the log-rank test statistic. Supports exact permutation test when the censoring distributions are equal in the treatment and the control group and approximate imputation-permutation methods when the censoring distributions are different.
Depends survival
License GPL-3 | file LICENSE
Encoding UTF-8
LazyData true
Imports stats, coin, utils
Suggests testthat
RoxygenNote 6.0.1
NeedsCompilation no
Author Matthias Brueckner [aut, cre],
     Martin Posch [aut],
     Franz Koenig [aut]
Repository CRAN
Date/Publication 2017-08-07 23:29:17 UTC

R topics documented:

createPermGS ............................................................... 2
exactLR ........................................................................... 3
imputeHeinze ................................................................. 4
imputeIPT ....................................................................... 4
imputeIPZ ..................................................................... 5
nextStage ................................................................. 6
parseFormula ............................................................... 7
**Description**

Create permGS object representing a permutational group-sequential trial.

**Usage**

```r
createPermGS(B = 1000, restricted = TRUE, method = "IPZ", pool = TRUE,
        type = c("logrank", "Gehan-Breslow", "Tarone-Ware", "Prentice",
                     "Self"), imputeData = NULL, permuteData = NULL)
```

**Arguments**

- `B` number of random permutations
- `restricted` if TRUE only permute within strata
- `method` imputation/permutation method IPZ, IPT, Heinze or none (default: IPZ)
- `pool` if TRUE impute event times from Kaplan-Meier estimator calculated from pooled data
- `type` logrank weights to be used with coin::logrank_trafo
- `imputeData` user-supplied imputation function (ignored if method is given)
- `permuteData` user-supplied permutation function (ignore if method is given)

**Value**

object of class permGS
**exactLR**

**Examples**

```r
## standard permutation test (no imputation, free permutations)
x <- createPermGS(1000, FALSE, "none")
summary(x)
## imputation using IPT method, restricted permutations
y <- createPermGS(1000, TRUE, "IPT")
summary(y)
```

**Description**

One-sided exact / approximate permutation and asymptotic log-rank test

**Usage**

```r
exactLR(B, formula, data = parent.frame(), type = "exact")
```

**Arguments**

- `B` number of random permutations (only used if type="approximate")
- `formula` a formula object, as used by `coxph`, left hand side must be a 'Surv' object, right hand side may only consist of a single term (treatment indicator)
- `data` data.frame or list containing the variables in "formula", by default "formula" is evaluated in the parent frame
- `type` if type="exact" performs complete enumeration of all permutations, if type="approximate" draw random permutations, if type="asymptotic" perform asymptotic log-rank test

**Details**

This function performs a standard exact or approximate permutation test which is only valid under the extended null hypothesis of equal survival AND censoring distributions.

**Value**

A list containing the exact or approximate permutation p-value and the observed test statistic

**Examples**

```r
T <- rexp(20)
C <- rexp(20)
data <- data.frame(time=pmin(T, C), status=(T<=C), trt=rbinom(20, 1, 0.5))

# Approximate permutation test using 1000 random permutations
x <- exactLR(1000, Surv(time, status) ~ trt, data, "approximate")
```
### imputeIPT

Impute data according to IPT method. Output is supposed to be passed to permute.IPT

**Usage**

```r
imputeIPT(data, pool = TRUE)
```

**Arguments**

- `data`: matrix as returned by `as.matrix(generateData(param))`
- `pool`: if TRUE impute events times from pooled Kaplan-Meier estimator (default: TRUE)

**Value**

list containing Kaplan-Meier estimators of censoring and survival distributions and the original data

**References**


---

### imputeHeinze

Impute data according to Heinze et al. method. Output is supposed to be passed to permute.heinze

**Usage**

```r
imputeHeinze(data, pool = TRUE)
```

**Arguments**

- `data`: matrix as returned by `as.matrix(generateData(param))`
- `pool`: if TRUE impute events times from pooled Kaplan-Meier estimator (default: TRUE)

**Value**

list containing Kaplan-Meier estimators of censoring and survival distributions and the original data

**References**

Arguments

data matrix as returned by as.matrix(generateData(param))

pool if TRUE impute events times from pooled Kaplan-Meier estimator (default: TRUE)

Value

matrix containing imputed survival and censoring times (columns 1 and 2), and original treatment indicator (column 3)

References


Description

Impute data according to IPZ method. Output is supposed to be passed to permute.IPZ

Usage

imputeIPZ(data, pool = TRUE)

Arguments

data matrix as returned by as.matrix(generateData(param))

pool if TRUE impute events times from pooled Kaplan-Meier estimator (default: TRUE)

Value

original data with 4 new columns (V1 and V2) containing the imputed observations

References

Description

Imputation permutation group-sequential log-rank test. Random permutations of a block are reused in all later stages. This automatically results in blockwise permutations.

Usage

```r
nextStage(pgs.obj, alpha, formula, data = parent.frame())
```

Arguments

- `pgs.obj`: permGS object as returned by `createpermgs`
- `alpha`: alpha at current stage
- `formula`: a formula object, as used by `coxph`, left hand side must be a 'Surv' object, right hand side must only consist of a factor (treatment indicator) and optionally a special `strata()` term identifying the permutation strata
- `data`: a data.frame or list containing the variables in "formula", by default "formula" is evaluated in the parent frame

Value

An updated permGS object.

Examples

```r
# Two-stage design with one-sided O'Brien-Fleming boundaries using IPZ method
x <- createPermGS(1000, TRUE, "IPZ")

# calendar time of interim analysis
T <- rexp(100) # event times
t1 <- 9
R <- runif(100, 0, 12) # recruitment times
t2 <- 18 # calendar time of final analysis
Z <- rbinom(100, 1, 0.5) # treatment assignment
C <- rexp(100) # drop-out times

# Stage 1 data
data.t1 <- data.frame(time=pmin(T, C, max(0, (t1-R))), status=(T<=pmin(C, t1-R)), trt=Z)
data.t1 <- data.t1[R <= t1]

# Stage 2 data
data.t2 <- data.frame(time=pmin(T, C, max(0, (t2-R))), status=(T<=pmin(C, t2-R)), trt=Z)
data.t2 <- data.t2[R <= t2]
x <- nextStage(x, 0.00153, Surv(time, status) ~ trt, data.t1)
summary(x)
```
if(!x$results$reject[1]) {
  data.t2$strata <- rep.int(c(1,2), c(nrow(data.t1), nrow(data.t2)-nrow(data.t1)))
  x <- nextStage(x, 0.025, Surv(time, status) ~ trt + strata(strata), data.t2)
  summary(x)
}

---

### parseFormula

**Parse formula of survival model**

**Description**

Parse formula of survival model

**Usage**

```r
parseFormula(formula, data = parent.frame())
```

**Arguments**

- `formula` : formula object
- `data` : data.frame (optional)

**Value**

data.frame containing the parsed variables

---

### permGS

**permGS**

**Description**

This package implements permutational group-sequential tests for time-to-event data based on (weighted) log-rank test statistics. It supports exact permutation test when the censoring distributions are equal in the treatment and the control group and the approximate imputation-permutation methods of Heinze et al. (2003) and Wang et al. (2010) and when the censoring distributions are different. Permutations can be stratified, i.e. only patients within the same stratum are treated as exchangeable. Rejection boundaries are monotone and finite even when only a random subset of all permutations is used. One- and Two-sided testing possible.

**Author(s)**

Matthias Brueckner <m.brueckner@lancaster.ac.uk>, Franz Koenig <Franz.Koenig@meduniwien.ac.at>, Martin Posch <martin.posch@meduniwien.ac.at>
References


Examples

```r
## IPZ method based on logrank test with 1000 restricted random permutations
x <- createPermGS(1000, TRUE, "IPZ", type="logrank")

T <- rexp(100) ## event times
R <- runif(100, 0, 12) ## recruitment times
Z <- rbinom(100, 1, 0.5) ## treatment assignment
C <- rexp(100) ## drop-out times

## two-stage design
T1 <- 9 ## calendar time of interim analysis
T2 <- 18 ## calendar time of final analysis

## Stage 1
data.t1 <- data.frame(time=pmin(T, C, max(0, (T1-R))), status=(T<=pmin(C, T1-R)), trt=Z)
data.t1 <- data.t1[R <= T1,]
x <- nextStage(x, 0.00153, Surv(time, status) ~ trt, data.t1)
summary(x)

if(!x$results$reject[1]) { ## Stage 2
data.t2 <- data.frame(time=pmin(T, C, max(0, (T2-R))), status=(T<=pmin(C, T2-R)), trt=Z)
data.t2 <- data.t2[R <= T2,]
data.t2$strata <- rep.int(c(1,2), c(nrow(data.t1), nrow(data.t2)-nrow(data.t1)))
x <- nextStage(x, alpha=0.025, Surv(time, status) ~ trt + strata(strata), data.t2)
summary(x)
}
```

---

**permHeinze**

Convenience function which calls createPermGS and nextStage to perform fixed sample size permutation test with Heinze method

**Description**

Convenience function which calls createPermGS and nextStage to perform fixed sample size permutation test with Heinze method
Usage

permHeinze(formula, data, B = 1000, alpha = 0.05, pool = TRUE, type = c("logrank", "Gehan-Breslow", "Tarone-Ware", "Prentice", "Prentice-Marek", "Andersen-Borgan-Gill-Keiding", "Fleming-Harrington", "Self"))

Arguments

- formula: a formula object, as used by `coxph`, left hand side must be a 'Surv' object, right hand side must only consist of a factor (treatment indicator) and optionally a special strata() term identifying the permutation strata.
- data: a data.frame or list containing the variables in "formula", by default "formula" is evaluated in the parent frame.
- B: number of random permutations (default: 1000).
- alpha: significance level (default: 0.05).
- pool: if TRUE impute event times from Kaplan-Meier estimator calculated from pooled data.
- type: logrank weights to be used with coin::logrank_trafo.

Value

An object of class permGS.

Examples

```r
T <- rexp(30) ## event times
Z <- rbinom(30, 1, 0.5) ## treatment assignment
C <- rexp(30) ## drop-out times
data <- data.frame(time=pmin(T,C), status=T<=C, Z=Z)
x <- permHeinze(Surv(time, status) ~ Z, data)
summary(x)
```

permIPT

Convenience function which calls createPermGS and nextStage to perform fixed sample size permutation test with IPT method.

Description

Convenience function which calls createPermGS and nextStage to perform fixed sample size permutation test with IPT method.

Usage

permIPT(formula, data, B = 1000, alpha = 0.05, pool = TRUE, type = c("logrank", "Gehan-Breslow", "Tarone-Ware", "Prentice", "Prentice-Marek", "Andersen-Borgan-Gill-Keiding", "Fleming-Harrington", "Self"))
Arguments

- **formula**: a formula object, as used by `coxph`, left hand side must be a `Surv` object, right hand side must only consist of a factor (treatment indicator) and optionally a special `strata()` term identifying the permutation strata.
- **data**: a data.frame or list containing the variables in "formula", by default "formula" is evaluated in the parent frame.
- **B**: number of random permutations (default: 1000)
- **alpha**: significance level (default: 0.05)
- **pool**: if TRUE impute event times from Kaplan-Meier estimator calculated from pooled data.
- **type**: logrank weights to be used with coin::logrank_trafo.

Value

An object of class `permGS`.

Examples

```r
T <- rexp(30)  ## event times
Z <- rbinom(30, 1, 0.5)  ## treatment assignment
C <- rexp(30)  ## drop-out times
data <- data.frame(time=pmin(T,C), status=T<=C, Z=Z)
x <- permIPZ(Surv(time, status) ~ Z, data)
summary(x)
```

Description

Convenience function which calls createPermGS and nextStage to perform fixed sample size permutation test with IPZ method.

Usage

```r
permIPZ(formula, data, B = 1000, alpha = 0.05, pool = TRUE,
         type = c("logrank", "Gehan-Breslow", "Tarone-Ware", "Prentice",
                  "Self"))
```
**Arguments**

- **formula**: a formula object, as used by `coxph`, left hand side must be a `Surv` object, right hand side must only consist of a factor (treatment indicator) and optionally a special `strata()` term identifying the permutation strata.
- **data**: a data.frame or list containing the variables in "formula", by default "formula" is evaluated in the parent frame.
- **B**: number of random permutations (default: 1000).
- **alpha**: significance level (default: 0.05).
- **pool**: if TRUE impute event times from Kaplan-Meier estimator calculated from pooled data.
- **type**: logrank weights to be used with coin::logrank_trafo.

**Value**

An object of class `permGS`.

**Examples**

```r
T <- rexp(30) ## event times
Z <- rbinom(30, 1, 0.5) ## treatment assignment
C <- rexp(30) ## drop-out times
data <- data.frame(time=pmin(T,C), status=T<=C, Z=Z)
x <- permr(formula=Surv(time, status) ~ Z, data)
summary(x)
```

**Description**

Convenience function which calls `createPermGS` and `nextStage` to perform fixed sample size permutation test without imputation.

**Usage**

```r
permr(formula, data, B = 1000, alpha = 0.05, pool = TRUE,
      type = c("logrank", "Gehan-Breslow", "Tarone-Ware", "Prentice",
               "Self"))
```
Arguments

formula a formula object, as used by `coxph`, left hand side must be a 'Surv' object, right hand side must only consist of a factor (treatment indicator) and optionally a special strata() term identifying the permutation strata
data a data.frame or list containing the variables in "formula", by default "formula" is evaluated in the parent frame
B number of random permutations (default: 1000)
alpha significance level (default: 0.05)
pool if TRUE impute event times from Kaplan-Meier estimator calculated from pooled data
type logrank weights to be used with coin::logrank_trafo

Value

An object of class permGS

Examples

```r
## Two-sided permutation test
t <- rexp(100) ## event times
z <- rbinom(100, 1, 0.5) ## treatment assignment
c <- rexp(100) ## drop-out times
data <- data.frame(time=pmin(t,c), status=t<=c, Z=z)
x <- permLR(Surv(time, status) ~ z, data, alpha=c(0.025, 0.025))
summary(x)
```

Description

Perform single imputation and permutation step

Usage

`permuteHeinze(imp, pp, index = TRUE)`

Arguments

imp list as returned by impute.heinze
pp vector of permuted indices
index not used

Value

matrix with time, status, trt columns
References


permuteIPT

Description
Permute survival times after imputation (IPT)

Usage
permuteIPT(data, pp, index = TRUE)

Arguments
- data: matrix as returned by impute.IPT
- pp: vector of permuted indices
- index: not used

Value
matrix with time, status, trt columns

References

permuteIPZ

Description
Permute treatment assignment after imputation (IPZ)

Usage
permuteIPZ(data, pZ, index = FALSE)
Arguments

data matrix as returned by impute.IPT
pZ vector of permuted indices if index is TRUE, else binary vector of treatment assignments
index indicates if pZ is a vector of indices or a binary vector of treatment assignments

Value

matrix with time, status, Z columns

References


Description

Sample from conditional distribution estimated by Kaplan-Meier estimator. Imputed values > tmax are right-censored.

Usage

sampleFromCondKM(u, fit, tmax = NULL, dv = 1, f = NULL)

Arguments

u vector of observed times
fit Kaplan-Meier fit as returned by survfit
tmax largest observation of the pooled sample
dv 1 if imputing events, 0 if imputing censoring times
f interpolated Kaplan-Meier estimate

Value

Random sample of survival times drawn from conditional distribution of T given T > U
**sampleFromKM**

**Description**

Sample from distribution estimated by Kaplan-Meier estimator. Imputed values > tmax are right-censored.

**Usage**

```r
sampleFromKM(n, fit, start = 0, tmax = NULL, dv = 1)
```

**Arguments**

- `n`: sample size
- `fit`: Kaplan-Meier fit as returned by `survfit`
- `start`: if 0 sample from L(T), else sample from L(T, T > start)
- `tmax`: largest observation in pooled sample
- `dv`: 1 if imputing events, 0 if imputing censoring times

**Value**

Random sample of survival times

**shuffleBlock**

**Description**

shuffleBlock Permute block preserving group sizes, randomization blocks

**Usage**

```r
shuffleBlock(block, strata = 0)
```

**Arguments**

- `block`: vector of row indices to be permuted
- `strata`: factor defining strata with block

**Value**

random permutation of each stratum within block
summary.permGS

summary.permGS  summary of permGS object

Description

summary of permGS object

Usage

## S3 method for class 'permGS'
summary(object, ...)

Arguments

  object  permGS object as returned by createPermGS

  ...  additional parameters (currently unused)

Value

  nothing
Index

coxph, 3, 6, 9–12
createPermGS, 2, 6, 16

exactLR, 3

imputeHeinze, 4
imputeIPT, 4
imputeIPZ, 5

nextStage, 6

parseFormula, 7
permGS, 7
permGS-package (permGS), 7
permHeinze, 8
permIPT, 9
permIPZ, 10
permLR, 11
permuteHeinze, 12
permuteIPT, 13
permuteIPZ, 13

sampleFromCondKM, 14
sampleFromKM, 15
shuffleBlock, 15
summary.permGS, 16