Package 'ICSKAT'

July 21, 2025

Type Package

Title Interval-Censored Sequence Kernel Association Test

Version 0.2.0

Description Implements the Interval-Censored Sequence Kernel Association (ICSKAT) test for testing the association between interval-censored time-to-event outcomes and groups of single nucleotide polymorphisms (SNPs). Interval-censored time-to-event data occur when the event time is not known exactly but can be deduced to fall within a given interval. For example, some medical conditions like bone mineral density deficiency are generally only diagnosed at clinical visits. If a patient goes for clinical checkups yearly and is diagnosed at, say, age 30, then the onset of the deficiency is only known to fall between the date of their age 29 checkup and the date of the age 30 checkup. Interval-censored data include right- and left-censored data as special cases. This package also implements the interval-censored Burden test and the ICSKATO test, which is the optimal combination of the ICSKAT and Burden tests. Please see the vignette for a quickstart guide.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.1

Imports CompQuadForm, dplyr, magrittr, Rcpp (>= 0.11.3), rje, survival, zoo

LinkingTo Rcpp, RcppEigen

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Ryan Sun [aut, cre], Liang Zhu [aut]

Maintainer Ryan Sun <ryansun.work@gmail.com>

Repository CRAN

Date/Publication 2021-11-25 06:00:02 UTC

Contents

`ACAT` *Aggregated Cauchy Association Test*

Description

A p-value combination method using the Cauchy distribution. Code provided by Dr. Yaowu Liu.

Usage

```
ACAT(Pvals, Weights = NULL)
```

Arguments

`Pvals`	a numeric vector of p-values to be combined by ACAT.
`Weights`	a numeric vector of non-negative weights for the combined p-values. When it is NULL, the equal weights are used.

Value

p-value of ACAT.

Author(s)

Yaowu Liu

Examples

```
p.values<-c(2e-02,4e-04,0.2,0.1,0.8)
ACAT(Pvals=p.values)
```

`calcScoreStats` *calcScoreStats.R*

Description

Function that is applied in ICsingleSNP() to calculate a score statistic and p-value for each column of an n*p genotype matrix.

Usage

```
calcScoreStats(x, UgTerm, ggTerm, gtTermCommon, gtHalfL, gtHalfR, solveItt)
```

Arguments

`x`	n*1 vector of genotypes.
`UgTerm`	n*1 vector multiplier for the score statistic.
`ggTerm`	n*1 vector multiplier for the Igg term of the variance.
`gtTermCommon`	n*p matrix multiplier for the common part of the Igt term of the variance.
`gtHalfL`	n*(nknots+1) matrix multiplier for one half of the unique part of the Igt term of the variance.
`gtHalfR`	n*(nknots+1) matrix multiplier for one half of the unique part of the Igt term of the variance.
`solveItt`	p*p inverse of the Itt matrix from ICSKAT_fit_null().

Value

A 2*1 vector with the test statistic and then p-value.

chiSqMatchFast *chiSqMatchFast.R*

Description

Match the moments of a mixture of scaled chi-square random variables to a single non-central chi-square, assumes the quadratic form case where the mean of the multivariate normal V=RV is 0.

Usage

```
chiSqMatchFast(lambdaVec, alwaysCentral = FALSE)
```

Arguments

lambdaVec	Numeric vector holding the eigenvalues of the A term, where we are interested in x^TAX and x is multivariate normal.
alwaysCentral	Boolean determining whether to always set the noncentrality parameter to 0, as in SKAT package.

Value

A list with the elements:

sigmaQrho	Standard deviation of the mixture distribution
muQrho	Mean of the mixture distribution
delta	Noncentrality parameter of the matched distribution
l	Degrees of freedom of the matched distribution

Examples

```
set.seed(2)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind, lt = lt, rt = rt)
icskatOut <- ICskat(left_dmat = dmats$left_dmat, right_dmat=dmats$right_dmat,
lt = lt, rt = rt, obs_ind = obs_ind, tpos_ind = tpos_ind, gMat = gMat,
null_beta = nullFit$beta_fit, Itt = nullFit$Itt)
```

```
Rrho <- matrix(data=0.5, nrow=20, ncol=20)
diag(Rrho) <- 1
toDecomp <- Rrho %*% icskatOut$sig_mat
tempEvals <- eigen(toDecomp, symmetric = TRUE, only.values = TRUE)$values
idx1 <- which(tempEvals >= 0)
idx2 <- which(tempEvals > mean(tempEvals[idx1])/100000)
tempEvals <- tempEvals[idx2]
chiSqMatchFast(lambdaVec = tempEvals)
```

`construct_interval_probs` *construct_interval_probs.R*

Description

Construct the probabilities of falling into each time interval for bootstrapping of interval-censored data.

Usage

```
construct_interval_probs(
  allTimes,
  dmats,
  nullBeta,
  p,
  nKnots,
  infVal = 999,
  zeroVal = 0
)
```

Arguments

`allTimes`	n*s matrix where n is number of subjects and s is all visit times for that subjects.
`dmats`	Output from make_IC_dmats, a list holding left_dmat and right_dmat.
`nullBeta`	Vector of coefficients under the null model.
`p`	Number of covariates in the null model.
`nKnots`	Number of knots in the spline.
`infVal`	The numeric value representing time 0 (left-censored observation).
`zeroVal`	The numeric value representing time infinity (right-censored observation).

Value

n*(s+1) matrix where element (i,j) holds the probability that subject i will fail in interval j.

Examples

```
set.seed(2)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind,
lt = lt, rt = rt)
intervalProbOutput <- construct_interval_probs(allTimes = outcomeDat$allVisits,
dmats = dmats, nullBeta = nullFit$beta_fit, p = ncol(xMat), nKnots=1)
```

---

coxphFn *coxphFn*

Description

Function to be applied over gMat to get p-values from coxPH().

Usage

```
coxphFn(x, xMat, midTime, midEvent, p)
```

Arguments

x	n*1 genotype vector.
xMat	n*p matrix of non-genotype covariates.
midTime	n*1 vector of event times imputed to be right-censored times using the midpoint imputation method.
midEvent	n*1 vector event indicators (0 for censoring, 1 for event) after times have been transformed to right-censored observations.
p	scalar, number of columns in xMat.

Value

A scalar p-value for testing the effect of the genotype in survreg() Weibull model.

createInt *Called by gen_IC_data() to turn the actual outcome times and observation times into interval-censored outcomes for each subject. Apply this with mapply over a data.frame of visit times, pass in the exact times.*

Description

Called by gen_IC_data() to turn the actual outcome times and observation times into interval-censored outcomes for each subject. Apply this with mapply over a data.frame of visit times, pass in the exact times.

Usage

```
createInt(obsTimes, eventTime)
```

Arguments

obsTimes	A vector of all the times a subject is observed.
eventTime	The exact event time for the subject.

Value

A 2*1 vector which is the interval of the event time

Examples

```
obsTimes <- 1:10
eventTime <- 7.7
createInt(obsTimes, eventTime)
```

fIntegrate *fIntegrate.R*

Description

The integrand in the SKATO p-value, pass it to a numerical integration function like integrate(), uses Davies method instead of Liu to calculate the probability in the integrand.

Usage

```
fIntegrate(x, muK1, sigmaK1, sigmaZeta, kappaLambda, QrhoDF)
```

Arguments

x	Variable to be integrated over, can be a vector or scalar.
muK1	Mean of the mixture of chi-squares that are the first part of the kappa variable. When we do bootstrap we often pass in the mean of the entire kappa, since the mean of zeta is supposed to be 0.
sigmaK1	Standard deviation of the entire kappa term.
sigmaZeta	Standard deviation of the zeta part of the kappa variable.
kappaLambda	Eigenvalues that weight the mixture of chi-squares that are the first part of the kappa variable.
QrhoDF	The data frame output from calling QrhoIC().

Value

The value of the integrand at x.

---

fIntegrateLiu *fIntegrateLiu.R*

Description

The integrand in the SKATO p-value when using Liu instead of Davies method, pass it to a numerical integration function like integrate().

Usage

```
fIntegrateLiu(x, muK1, sigmaK1, QrhoDF, dfK1)
```

Arguments

x	Variable to be integrated over, can be a vector or scalar.
muK1	Mean of the mixture of chi-squares that are the first part of the kappa variable.
sigmaK1	Standard deviation of the mixture of chi-squares that are the first part of the kappa variable.
QrhoDF	The data frame output from calling QrhoIC().
dfK1	The degrees of freedom from the approximated chi-square.

Value

The value of the integrand at x.

gen_IC_data *gen_IC_data.R*

Description

Generate interval-censored data under the proportional odds/PH model given a baseline hazard function and some information about observation times.

Usage

```
gen_IC_data(bhFunInv, obsTimes, windowHalf, etaVec, mod = "PH", probMiss = 0.1)
```

Arguments

bhFunInv	A function, the inverse of the baseline hazard function.
obsTimes	Vector of the intended observation times.
windowHalf	The amount of time before or after the intended obsTimes that a visit might take place.
etaVec	n*1 linear predictor in either the proportional odds or proportional hzards model.
mod	Either "PH" to generate under PH model or "PO" to generate under PO model.
probMiss	The probability of missing any given visit.

Value

A list with the elements:

obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
tVec	Fisher information matrix for the fitted coefficients.
leftTimes	n*1 vector of left side of interval times.
rightTimes	n*1 vector of right side of interval times.
tVec	n*1 vector of exact event times.

Examples

```
set.seed(0)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
```

ICsingleSNP *ICsingleSNP.R*

Description

Burden test from ICSKAT() except do a separate burden test for each SNP in gMat, one at a time.

Usage

```
ICsingleSNP(
  left_dmat,
  right_dmat,
  lt,
  rt,
  obs_ind,
  tpos_ind,
  gMat,
  null_beta,
  solveItt,
  p
)
```

Arguments

left_dmat	n*(p+nknots+2) design matrix for left end of interval.
right_dmat	n*(p+nknots+2) design matrix for right end of interval.
lt	n*1 vector of left side of time interval.
rt	n*1 vector of right side of time interval.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
gMat	n*q genotype matrix.
null_beta	(p+nknots+2)*1 vector of coefficients for null model.
solveItt	Inverse of (p+nknots+2)*(p+nknots+2) Fisher information matrix for null model coefficients.
p	number of non-SNP covariates.

Value

A list with the elements:

testStatsVec	p*1 vector of score test statistics
pVec	p*1 vector of score test p-values

Examples

```
set.seed(0)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind, lt = lt, rt = rt)
solveItt  <- solve(nullFit$Itt)
ICsingleSNP(left_dmat = dmats$left_dmat, right_dmat=dmats$right_dmat, lt = lt, rt = rt,
obs_ind = obs_ind, tpos_ind = tpos_ind, gMat = gMat, null_beta = nullFit$beta_fit,
solveItt = solveItt, p=2)
```

ICskat *ICSKAT.R*

Description

Calculate the test statistic and p-value for interval-censored SKAT.

Usage

```
ICskat(
  left_dmat,
  right_dmat,
  lt,
  rt,
  obs_ind,
  tpos_ind,
  gMat,
  null_beta,
  Itt,
  pvalue = TRUE
)
```

Arguments

left_dmat	n*(p+nknots+2) design matrix for left end of interval.
right_dmat	n*(p+nknots+2) design matrix for right end of interval.

lt	n*1 vector of left side of interval times.
rt	n*1 vector of right side of interval times.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
gMat	n*q genotype matrix.
null_beta	(p+nknots+2)*1 vector of coefficients for null model.
Itt	(p+nknots+2)*(p+nknots+2) Fisher information matrix for null model coefficients.
pvalue	Boolean, if TRUE then find the p-value (maybe don't need it if bootstrapping, saves eigendecomposition)

Value

A list with the elements:

p_SKAT	ICSKAT p-value
p_burden	IC burden test p-value
complex	Indicator of whether the SKAT variance matrix was positive definite
sig_mat	The covariance matrix of the score equations for genetic effects when treated as fixed effects
skatQ	SKAT test statistic
burdenQ	Burden test statistic
Ugamma	Score vector
lambdaQ	Vector of eigenvalues of variance matrix
null_beta	The fitted null parameters
err	Will be 0 for no error, 22 if had to adjust parameters on CompQuadForm (totally normal), or 99 if NA in variance matrix. ICSKATwrapper will return 1 here if the null fit has an error
errMsg	Explains error code, blank string if no error

Examples

```
set.seed(2)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
```

```
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind,
lt = lt, rt = rt)
ICskat(left_dmat = dmats$left_dmat, right_dmat=dmats$right_dmat,
lt = lt, rt = rt, obs_ind = obs_ind, tpos_ind = tpos_ind, gMat = gMat,
null_beta = nullFit$beta_fit, Itt = nullFit$Itt)
```

ICSKATO *ICSKATO.R*

Description

Calculate SKATO test for ICSKAT.

Usage

```
ICSKATO(
  rhoVec = c(0, 0.01, 0.04, 0.09, 0.25, 0.5, 1),
  icskatOut,
  useMixtureKurt = FALSE,
  liu = TRUE,
  liuIntegrate = FALSE,
  bootstrapOut = NULL,
  alwaysCentral = FALSE,
  ACAT = FALSE
)
```

Arguments

rhoVec	Vector of rhos to search over.
icskatOut	The output list from ICSKAT().
useMixtureKurt	Boolean for whether to use the mixture formula to estimate the kurtosis of Qrho when we have bootstrap results. Default is false, instead we just use the bootstraped kurtosis of Qrho.
liu	Boolean for whether to use Liu moment matching approximation for p-value of each Qrho (as opposed to Davies). If Davies, cannot use bootstrapped moments of Qrho.
liuIntegrate	Boolean for whether to use Liu moment matching approximation integration in SKATO p-value (as opposed to Davies).
bootstrapOut	Output list from call to ICSKATO_bootstrap().
alwaysCentral	A boolean, if TRUE, follow SKAT package practice of always setting delta=0 in chi-square moment matching.
ACAT	Uses the ACAT method to perform ICSKATO, will result in a conservative test but is much faster.

Value

A list with the elements:

pval	SKATO p-value.
correctedP	Corrected SKATO p-value, which will be the same as pval when not all Qrho values produce a p-value between 0 and 1 (e.g. sometimes it will be 0). Correction is same as SKAT package correction..
QrhoDF	Data frame containing the distribution and p-value for each Qrho.
r	The rank of the cholesky decomposition of the sig_mat returned from ICSKAT(), i.e. V^-1/2 or Z.
intDavies	Boolean denoting whether integration was with Davies (true) or Liu method (false).
err	0 is no error, 1 is early error like possibly only one eigenvalue/issue with sigmat/issue with kappaMat/issue with QrhoDF, 2 is corrected p-value (fine), 3 is integration error, 9 is no positive p-values (so SKATOp should be 0 unless burden is 1).
lambdaKurtK1	Kurtosis of kappa term minus zeta using eigenvalues, we use it to approximate the kurtosis of the entire kappa.
lambdaSigmaK1	Standard deviation of kappa term, including zeta, using eigenvalues.
lambdaMuK1	Mean of kappa term using eigenvalues.
bootKurtKappaAll	Kurtosis of entire kappa term, including zeta, using bootstrap data
bootSigmaKappaAll	Standard deviation of entire kappa term using bootstrap data.
bootMuKappaAll	Mean of entire kappa term using bootstrap data.
mixDFVec	Degrees of freedom of Qrho if useMixtureKurt is true, only here to match SKAT package, not really used.

Examples

```
set.seed(1)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind,
lt = lt, rt = rt)
```

```
icskatOut <- ICskat(left_dmat = dmats$left_dmat, right_dmat=dmats$right_dmat,
lt = lt, rt = rt, obs_ind = obs_ind, tpos_ind = tpos_ind, gMat = gMat,
null_beta = nullFit$beta_fit, Itt = nullFit$Itt)
ICSKATO(icskatOut = icskatOut)
```

ICSKATO_bootstrap *ICSKATO_bootstrap.R*

Description

The version of ICSKATO to run when bootstrapping to match kurtosis.

Usage

```
ICSKATO_bootstrap(
  icskatOut,
  B,
  intervalProbs,
  allVisits,
  quant_r,
  seed = NULL,
  null_fit,
  gMat,
  xMat,
  fitAgain,
  checkpoint = FALSE,
  downsample = 1,
  rhoVec
)
```

Arguments

icskatOut	The output list from ICSKAT().
B	Number of bootstrap replications.
intervalProbs	n*(s+1) matrix where n is number of subjects and s is the number of visits possible, probability of falling in each interval.
allVisits	n*s matrix with all the visit times for each subject.
quant_r	Quantiles of time from make_IC_dmats, to keep them constant through bootstrapping.
seed	Seed to start the bootstrapping.
null_fit	The null fit output from ICSKAT_fit_null.
gMat	Genotype matrix used in original test.
xMat	n*p matrix of non-genetic covariates.
fitAgain	Boolean, whether to fit the null model again in each bootstrap.

checkpoint	Boolean, whether to print every time 100 bootstraps finish.
downsample	A number in (0, 1], will use this fraction of the bootstrap iterations to try running the test with fewer bootstraps.
rhoVec	Vector of rhos to search over in SKATO.

Value

A list with the elements:

kurtQvec	Vector of bootstrapped excess kurtosis of each Qrho.
varQvec	Vector of bootstrapped variance of each Qrho.
meanQvec	Vector of bootstrapped mean of each Qrho.
kurtKappa	Bootstrapped kurtosis of kappa term without zeta.
kurtKappaAll	Bootstrapped kurtosis of full kappa term with zeta.
varKappaAll	Bootstrapped variance of full kappa term with zeta.
meanKappaAll	Bootstrapped mean of full kappa term with zeta.
bootDF	Matrix with B rows containing all the bootstrapped quantities over all iterations.
QrhoBoot	Matrix with B rows containing all the bootstrapped Qrho values, one column for each rho.
listDS	A list containing all of the other elements in this return list, except using the downsampled iterations.
nonNA	Number of bootstraps that did not result in NA (and thus were not removed).

Examples

```
set.seed(2)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind,
lt = lt, rt = rt)
icskatOut <- ICskat(left_dmat = dmats$left_dmat, right_dmat=dmats$right_dmat,
lt = lt, rt = rt, obs_ind = obs_ind, tpos_ind = tpos_ind, gMat = gMat,
null_beta = nullFit$beta_fit, Itt = nullFit$Itt)
intervalProbOutput <- construct_interval_probs(allTimes = outcomeDat$allVisits,
dmats = dmats, nullBeta = nullFit$beta_fit, p = ncol(xMat), nKnots=1)
ICSKATO_bootstrap(icskatOut = icSkatOut, B = 100, intervalProbs = intervalProbOutput$probMat,
```

```
allVisits = intervalProbOutput$allTimesFilled, quant_r = dmats$quant_r, seed = 0,
null_fit = nullFit, gMat = gMat, xMat, fitAgain = TRUE,
rhoVec=c(0, 0.01, 0.04, 0.09, 0.25, 0.5, 1))
```

ICskatPO *ICSKATPO.R*

Description

Calculate the test statistic and p-value for interval-censored skat with PO model.

Usage

```
ICskatPO(
  left_dmat,
  right_dmat,
  lt,
  rt,
  obs_ind,
  tpos_ind,
  gMat,
  null_beta,
  Itt
)
```

Arguments

left_dmat	n*(p+nknots+2) design matrix for left end of interval.
right_dmat	n*(p+nknots+2) design matrix for right end of interval.
lt	n*1 vector of left side of interval times.
rt	n*1 vector of right side of interval times.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
gMat	n*q genotype matrix.
null_beta	(p+nknots+2)*1 vector of coefficients for null model.
Itt	(p+nknots+2)*(p+nknots+2) Fisher information matrix for null model coefficients.

Value

A list with the elements:

p_SKAT	ICSKAT p-value for PO model.
p_burden	IC burden test p-value for PO model.
complex	Indicator of whether the SKAT variance matrix was positive definite

sig_mat	The covariance matrix of the score equations for genetic effects when treated as fixed effects
skatQ	SKAT test statistic.
burdenQ	Burden test statistic.
err	err=1 for a bad null fit.
errMsg	Describes the error.

Examples

```
set.seed(0)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0.1, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind, lt = lt, rt = rt)
ICskatPO(left_dmat = dmats$left_dmat, right_dmat=dmats$right_dmat, lt = lt, rt = rt,
obs_ind = obs_ind, tpos_ind = tpos_ind, gMat = gMat, null_beta = nullFit$beta_fit,
Itt = nullFit$Itt)
```

ICSKATwrapper *ICSKATwrapper.R*

Description

Wrapper to fit the null model and run ICSKAT all in one instead of separately - offers some functionality for error checking or using different initial values when fit fails to converge.

Usage

```
ICSKATwrapper(
  left_dmat,
  right_dmat,
  initValues,
  lt,
  rt,
  obs_ind,
  tpos_ind,
```

```
  gMat,
  PH = TRUE,
  nKnots = 1,
  maxIter = 3,
  eps = 10^(-6),
  runOnce = FALSE,
  returnNull = FALSE
)
```

Arguments

left_dmat	n*(p+nknots+2) design matrix for left end of interval.
right_dmat	n*(p+nknots+2) design matrix for right end of interval.
initValues	(p+nknots+2)*1 vector of coefficients to initialize the Newton-Raphson.
lt	Left side of interval times.
rt	Right side of interval times.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
gMat	n*q matrix of genotypes.
PH	Boolean for whether to fit PH model (TRUE) or PO model (FALSE).
nKnots	Number of knots in the spline.
maxIter	Number of times to try the fit if initial values do not lead to convergence.
eps	Difference in L2 norm of fitted null coefficients that stops the Newton Raphson.
runOnce	Boolean, if true then just go through the algorithm once with the initial values for coefficients, updating the variance matrix, useful for bootstrapping.
returnNull	Return a list with the skat output and null model, or just return the skat output (FALSE).

Value

Either a list with skatOutput and nullFit (two lists), or just skatOutput.

Examples

```
set.seed(0)
gMat <- matrix(data=rbinom(n=2000, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
```

```
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
ICSKATwrapper(left_dmat = dmats$left_dmat, right_dmat = dmats$right_dmat,
initValues = rep(0, ncol(xMat) + 3), lt = lt, rt = rt, obs_ind = obs_ind,
tpos_ind = tpos_ind, gMat = gMat, returnNull = TRUE)
```

ICSKAT_fit_null *ICSKAT_fit_null.R*

Description

Fit the null model (cubic basis spline for baseline cumulative hazard and coefficients for non-genetic coefficiens) for interval-censored skat.

Usage

```
ICSKAT_fit_null(
  init_beta,
  left_dmat,
  right_dmat,
  obs_ind,
  tpos_ind,
  lt,
  rt,
  runOnce = FALSE,
  checkpoint = FALSE,
  eps = 10^(-6)
)
```

Arguments

init_beta	(p+nknots+2)*1 vector of coefficients to initialize the Newton-Raphson.
left_dmat	n*(p+nknots+2) design matrix for left end of interval.
right_dmat	n*(p+nknots+2) design matrix for right end of interval.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
lt	n*1 vector of left interval times.
rt	n*1 vector of right interval times.
runOnce	Boolean tells the function to just go through the loop once instead of converging (to get quantites for bootstrapping).
checkpoint	Boolean tells the function to print when each iteration completes.
eps	Stop when the L2 norm of the difference in model coefficients reaches this limit.

Value

A list with the elements:

beta_fit	(p+nknots+2)*1 vector of fitted coefficients under null model.
iter	Number of iterations needed to converge.
Itt	Fisher information matrix for the fitted coefficients.
diff_beta	Difference between beta_fit and previous iteration of the vector, can be checked for errors.
err	Value is 0 if no errors and 1 if Itt is singular, can't perform fit.
errMsg	Empty string if err=0, explains error if there is one.

Examples

```
set.seed(2)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind, lt = lt, rt = rt)
```

ICSKAT_fit_null_PO *ICSKAT_fit_null_PO.R*

Description

Fit the null model (cubic basis spline for baseline cumulative hazard and coefficients for non-genetic coefficients) for interval-censored skat with PO model.

Usage

```
ICSKAT_fit_null_PO(
  init_beta,
  left_dmat,
  right_dmat,
  obs_ind,
  tpos_ind,
  lt,
  rt,
```

```
  checkpoint = FALSE,
  eps = 10^(-6)
)
```

Arguments

init_beta	(p+nknots+2)*1 vector of coefficients to initialize the Newton-Raphson.
left_dmat	n*(p+nknots+2) design matrix for left end of interval.
right_dmat	n*(p+nknots+2) design matrix for right end of interval.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
lt	n*1 vector of left side of interval times.
rt	n*1 vector of right side of interval times.
checkpoint	Boolean tells the function to print when each iteration completes.
eps	Stop when the L2 norm of the difference in model coefficients reaches this limit.

Value

A list with the elements:

beta_fit	(p+nknots+2)*1 vector of fitted coefficients under null model.
iter	Number of iterations needed to converge.
Itt	Fisher information matrix for the fitted coefficients.
diff_beta	Difference between beta_fit and previous iteration of the vector, can be checked for errors.
err	err=1 if NA shows up in the calculation.
IterrMsg	Describes the error.

Examples

```
set.seed(2)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat = xMat, lt = lt, rt = rt, obs_ind = obs_ind,
tpos_ind = tpos_ind)
ICSKAT_fit_null_PO(init_beta = rep(0.1, 5), left_dmat = dmats$left_dmat,
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind, lt = lt, rt = rt)
```

make_IC_dmat *make_IC_dmat.R*

Description

Puts together the entire design matrix for both the left and right ends of the interval, pasting together the non-genetic covariates with the cubic spline basis.

Usage

```
make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind, quant_r = NULL, nKnots = 1)
```

Arguments

xMat	n*p matrix of non-genetic covariates.
lt	n*1 vector with left end of intervals (min is 0).
rt	n*1 vector with right end of intervals.
obs_ind	n*1 vector of whether the event was observed before last follow-up.
tpos_ind	n*1 vector of whether the event was observed after follow-up started (t>0).
quant_r	Quantiles of time to use in constructing the spline, pass in if doing bootstrap.
nKnots	Number of knots to use for cubic spline basis (default is 1).

Value

A list with the elements:

right_dmat	n*(p+nKnots+2) design matrix for right end of interval.
left_dmat	n*(p+nKnots+2) design matrix for left end of interval.
quant_r	Quantiles used for constructing spline.

Examples

```
set.seed(0)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
make_IC_dmat(xMat = xMat, lt = lt, rt = rt, obs_ind = obs_ind, tpos_ind = tpos_ind)
```

`matchVisit` *match_visit.R*

Description

Match visit to a time for model-based bootstrap with interval-censored data.

Usage

```
matchVisit(draw, visitTimes)
```

Arguments

`draw`	(s+1)-length vector of all 0s except for one 1, which is the failure interval.
`visitTimes`	s-length vector where is the number of inspection times for a subject.

Value

n*(s+1) matrix where element (i,j) holds the probability that subject i will fail in interval j.

`mixture_kurtosis` *mixture_kurtosis.R*

Description

Calculate the kurtosis of Qrho when performing SKATO with bootstrapped moments. This function is included to allow for the potential to match the SKAT package, however we generally don't call it because we can just bootstrap the kurtosis of Qrho directly if we are already doing bootstrap, thus avoiding this calculation. Also it's only used in calculating the qmin values, not in the final p-value calculation, which uses a kappa expression that is only the first two terms of Qrho.

Usage

```
mixture_kurtosis(tempDF1, tempDF2, v1, a1, a2)
```

Arguments

`tempDF1`	Generally the bootstrapped kurtosis of the mixture of chi-squares in kappa.
`tempDF2`	Generally 1 because it's for the chi-square1 RV in kappa.
`v1`	Generally the variance of the mixture of chi-squares plus the variance of zeta in kappa.
`a1`	Generally the 1-rho in front of the first part of the kappa term.
`a2`	Generally the tau(rho) term in front of the chi-square1 RV in kappa.

Value

Kurtosis (excess kurtosis to be more precise), use df = 12 / kurtosis.

QrhoIC *QrhoIC.R*

Description

Calculate the test statistic, distribution, and p-value for each value of Krho in SKATO.

Usage

```
QrhoIC(
  rhoVec,
  icskatOut,
  liu = TRUE,
  bootstrapOut = NULL,
  alwaysCentral = FALSE
)
```

Arguments

rhoVec	Numeric vector of the rho values to use in SKATO.
icskatOut	The output list returned from a call to ICSKAT().
liu	Boolean for whether to use Liu (TRUE) or Davies (FALSE) method in calculating p-values for each Qrho. Default is Liu, following SKAT package. If wanting to use bootstrap moments for Qrho, need to use Liu method.
bootstrapOut	The output (a list) from a call the ICSKATO_bootstrap() function, holding moments for Qrho.
alwaysCentral	A boolean, if TRUE, follow SKAT package practice of always setting delta=0 in chi-square moment matching.

Value

Data frame holding the SKAT pvalue + test statistic for each fixed rho, the matched noncentrality + degree of freedom parameters for each fixed rho (using both bootstrap and analytic calculation), and the mean and variance of each Qrho using both bootstrap and analytic calculation.

singleSNPalt *singleSNPalt.R*

Description

Take a matrix of SNPs and get the interval-censored regression p-value for each one separately using either survreg() or coxph() with midpoint approximation.

Usage

```
singleSNPalt(
  lt,
  rt,
  tpos_ind,
  obs_ind,
  xMat,
  gMat,
  coxph = TRUE,
  survreg = TRUE
)
```

Arguments

lt	n*1 vector of left side of time interval.
rt	n*1 vector of right side of time interval.
tpos_ind	n*1 binary vector of whether the event was observed after follow-up started (takes value 1 if t>0, 0 otherwise).
obs_ind	n*1 vector of whether the event was observed or right-censored (takes value 1 if observed or 0 if right-censored).
xMat	non-SNP covariates matrix.
gMat	n*q genotype matrix.
coxph	Boolean, whether to fit Cox PH model.
survreg	Boolean, whether to fit survreg() Wiibull model.

Value

A list with the elements:

pvalCox	q*1 vector of marginal SNP p-values with Cox model
pvalSurv	q*1 vector of marginal SNP p-values with survreg Weibull model

Examples

```
set.seed(2)
gMat <- matrix(data=rbinom(n=200, size=2, prob=0.3), nrow=100)
xMat <- matrix(data=rnorm(200), nrow=100)
bhFunInv <- function(x) {x}
obsTimes <- 1:5
etaVec <- rep(0, 100)
outcomeDat <- gen_IC_data(bhFunInv = bhFunInv, obsTimes = obsTimes, windowHalf = 0.1,
probMiss = 0.1, etaVec = etaVec)
lt <- outcomeDat$leftTimes
rt <- outcomeDat$rightTimes
tpos_ind <- as.numeric(lt > 0)
obs_ind <- as.numeric(rt != Inf)
dmats <- make_IC_dmat(xMat, lt, rt, obs_ind, tpos_ind)
nullFit <- ICSKAT_fit_null(init_beta = rep(0, 5), left_dmat = dmats$left_dmat,
```

```
right_dmat=dmats$right_dmat, obs_ind = obs_ind, tpos_ind = tpos_ind, lt = lt, rt = rt)
singleSNPalt(lt = lt, rt = rt,tpos_ind = tpos_ind, obs_ind = obs_ind, xMat = xMat, gMat = gMat)
```

`survregFn` *survregFn*

Description

Function to be applied over gMat to get p-values from survreg().

Usage

```
survregFn(x, xMat, leftTime2, rightTime2, p)
```

Arguments

`x`	n*1 genotype vector.
`xMat`	n*p matrix of non-genotype covariates.
`leftTime2`	n*1 vector of left interval times in the format of Surv() interval2 type, i.e NA for left or right censored observations.
`rightTime2`	n*1 vector of right interval times in the format of Surv() interval2 type, i.e NA for left or right censored observations.
`p`	scalar, number of columns in xMat.

Value

A scalar p-value for testing the effect of the genotype in survreg() Weibull model.

Index