Package ‘stepp’

June 3, 2024

Type Package

Title Subpopulation Treatment Effect Pattern Plot (STEPP)

Version 3.2.7

Date 2024-06-02

Author Wai-ki Yip [aut, cre],
Ann Lazar [ctb],
David Zahrieh [ctb],
Chip Cole [ctb],
Ann Lazar [ctb],
Marco Bonetti [ctb],
Victoria Wang [ctb],
William Barcella [ctb],
Sergio Venturini [aut]
Richard Gelber [ctb]

Maintainer Wai-ki Yip <yuser86@yahoo.com>

Imports rstudioapi, scales

Depends methods, car, survival, splines

Collate 'zzz.R' 'utility.R' 'stutil.R' 'stwin.R' 'stsubpop.R'
'stmodelKM.R' 'stmodelCI.R' 'stmodelGLM.R' 'stmodel.R'
'steppes.R' 'stepp.R'

Description A method to explore the treatment-covariate interactions in survival or generalized linear model (GLM) for continuous, binomial and count data arising from two or more treatment arms of a clinical trial. A permutation distribution approach to inference is implemented, based on permuting the covariate values within each treatment group.

License GPL (>= 2)

URL https://www.r-project.org

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-06-03 02:20:02 UTC

R topics documented:

`analyze.CumInc.stepp` *Analyze competing risks data using Cumulative Incidence method*

Description

This method will be deprecated in the future. Please use the constructor function stepp.CI to create a STEPP Cumulative Incidence model for future development.

A method to explore the treatment-effect interactions in competing risks data arising from two treatment arms of a clinical trial. A permutation distribution approach to inference is implemented that permutes covariate values within a treatment group. The statistical significance of observed heterogeneity of treatment effects is calculated using permutation tests: 1) for the maximum difference between each subpopulation effect and the overall population treatment effect or supremum based test statistic; 2) for the difference between each subpopulation effect and the overall population treatment effect, which resembles the chi-square statistic.

Usage

```
analyze.CumInc.stepp(coltrt, coltime, coltype, covar, trts, patspop, minpatspop,
   timest, noperm=2500, ncex = 0.7, legendy = 30, pline = -2.5,
   color = c("red", "black"),
   xlabel = "Subpopulations by Median Covariate",
   ylabel = "?-year Disease-Free Survival",
   tlegend = c("1st Treatment", "2nd Treatment"),
   nlas = 3, pointwise = FALSE)
```

Arguments

`coltrt`	the treatment variable
`coltime`	the time to event variable
`coltype`	variable with distinct codes for different causes of failure where coltype=0 for censored observations; coltype=1 for event of interest; coltype=2 for other causes of failure
`covar`	the covariate of interest
`trts`	a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment arms, respectively
`patspop`	larger parameter(r2) for subpopulation construction that determines how many patients are in each subpopulation
`minpatspop`	smaller parameter(r1) for subpopulation construction that determines the largest number of patients in common among consecutive subpopulations
`timest`	timepoint to estimate survival
`noperm`	the desired number of permutations; must be 0 or above
`ncex`	optional - specify the size of the text for the sample size annotation, that is, the character expansion factor
`legendy`	optional - the vertical location of the legend according to the units on the y-axis
`pline`	optional - specify the vertical location of the p-value, that is, on which margin line, starting at 0 counting outwards
`color`	optional - a vector containing the line colors for the 1st and 2nd treatment, respectively
`ylabel`	optional - specify the label for the y-axis
`xlabel`	optional - specify the label for the x-axis
`tlegend`	optional - a vector containing the treatment labels, 1st and 2nd treatment, respectively
`nlas`	optional - specify the las parameter (0,1,2,3) to determine the orientation of the sample size annotation
`pointwise`	optional -specify pointwise confidence intervals (pointwise=TRUE), or confidence bands (pointwise=FALSE, default) to be displayed

Details

A statistical method to explore treatment by covariate interactions in competing risks data arising from two treatment arms of a clinical trial. The method is based on constructing overlapping subpopulations of patients with respect to a covariate of interest, and in observing the pattern of the treatment effects estimated across subpopulations. A plot of these treatment effects is called STEPP, or Subpopulation Treatment Effect Pattern Plot. STEPP uses the permutation distribution based approach for inference.

One can explore the window parameters without invoking the permutation analysis by setting noperm to 0. In that case, pvalue and the covarance matrix will not be available.

We acknowledge Robert J. Gray for permitting us to use the cmprsk package.

Value

analyze.CumInc.stepp generates a Subpopulation Treatment Effect Pattern Plot (STEPP) displaying the p-value from the test for interaction. Descriptive summaries of the dataset and the estimated variance-covariance matrix are returned through the steppes object. See documentation on steppes object for details on how you can use it.

Warning

This function together with other old functions will be depreciated in the future. A new set of S4 classes are implemented to replace old interfaces. Please use them for future development.

A few tips to keep in mind:

The variables coltrt, coltime, coltype, and covar must be numeric. No formatting allowed.

If you receive the error "Error in solve.default(sigma): system is computationally singular; reciprocal condition number = 0" then we recommend changing the seed by re-running analyze.KM.stepp. If this error persists after several runs, then the program cannot provide reliable results. Please try modifying your choices of the two parameters minpatspop(r1) and patspop(r2) that define the subpopulation.

The number of permutations specified in noperm, the sample size, and the number of subpopulations generated will affect how long analyze.CumInc.stepp takes to execute. The results are stable if 2500 or more permutations are specified. Furthermore, varying the number of subpopulations will affect inference.

The time point selected to estimate survival in timest must be in the same units (e.g., months) as the coltime variable.

The order of the treatments in the vector trts must be in the same order in the vector tlegend.

Note

STEPP is an exploratory tool, with graphical features that make it easy for clinicians to interpret the results of the analysis. Positive results should prompt the need for confirmation from other datasets investigating similar treatment comparisons. It should also be clear that STEPP is not meant to estimate specific cutpoints in the range of values of the covariate of interest, but rather to provide some indication on ranges of values where treatment effect might have a particular behavior.

STEPP considers the case in which the subpopulations are constructed according to a sliding window pattern. The larger parameter (patspop) determines how many patients are in each subpopulation, and the smaller parameter (minpatspop) determines the largest number of patients in common among consecutive subpopulations. A minimum of 80-100 patients should be in each subpopulation, but that is not strictly necessary. The difference (patspop-minpatspop) is the approximate number of patients replaced between any two subsequent subpopulations, and can be used to determine the number of subpopulations once patspop is fixed. The choice of the values of the parameters patspop and minpatspop to be used does change the appearance of the plot and the corresponding p-value. It is probably reasonable to experiment with a few combinations to ensure that the significance (or lack of significance) is stable with respect to that choice.

For best results, consider implementing 2500 permutations of the covariate (vector of subpopulations) to obtain a rich distribution in which to draw inference.

Author(s)

Ann Lazar, Wai-ki Yip, David Zahrieh, Bernard Cole, Marco Bonetti, Richard Gelber

References

Bonetti M, Gelber RD. Patterns of treatment effects in subsets of patients in clinical trials. Biostatistics 2004; 5(3):465-481.

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

Lazar AA, Cole BF, Bonetti M, Gelber RD. Evaluation of treatment-effect heterogeneity usiing biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot. Journal of Clinical Oncology, 2010; 28(29): 4539-4544.

See Also

`stwin`, `stsubpop`, `stmodelCI`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`,

Old functions to be deprecated: stepp, stepp_summary, stepp_print, stepp_plot, and analyze.KM.stepp.

Examples

```
#GENERATE TREATMENT VARIABLE:
n <- 1000 # set the sample size
mu <- 0 # set the mean and sd of the covariate
sigma <- 1

beta0 <- log(-log(0.5)) # set the intercept for the log hazard
beta1 <- -0.2 # set the slope on the covariate
beta2 <- 0.5 # set the slope on the treatment indicator
beta3 <- 0.7 # set the slope on the interaction

prob2 <- 0.2 # set the proportion type 2 events
cprob <- 0.3 # set the proportion censored

set.seed(7775432)  # set the random number seed
```

```
covariate <- rnorm(n,mean=mu,sd=sigma) # generate the covariate values
Txassign <- rbinom(n,1,0.5) # generate the treatment indicator
x3 <- covariate*Txassign # compute interaction term
# compute the hazard for type 1 event
lambda1 <- exp(beta0+beta1*covariate+beta2*Txassign+beta3*x3)
lambda2 <- prob2*lambda1/(1-prob2) # compute the hazard for the type 2 event
# compute the hazard for censoring time
lambda0 <- cprob*(lambda1+lambda2)/(1-cprob)
t1 <- rexp(n,rate=lambda1) # generate the survival time for type 1 event
t2 <- rexp(n,rate=lambda2) # generate the survival time for type 2 event
t0 <- rexp(n,rate=lambda0) # generate the censoring time
time <- pmin(t0,t1,t2) # compute the observed survival time
type <- rep(0,n)
type[(t1 < t0)&(t1 < t2)] <- 1
type[(t2 < t0)&(t2 < t1)] <- 2

# Call analyze.CumInc.stepp to analyze the data
# Warning: In this example, the permutations have been set to 0 to allow the stepp function
# to finish in a short amount of time.  IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO
# PROVIDE STABLE RESULTS.
analyze.CumInc.stepp(coltrt=Txassign, trts=c(0,1), coltime=time, coltype=type, covar=covariate,
  patspop=300, minpatspop=200, timest=1.0, noperm=0,
  ncex=0.70,legendy=30,pline=-2.5,color=c("red", "black"),
  xlabel="Subpopulations by Median Age",ylabel="4-year Cancer Relapse",
  tlegend=c("Treatment A", "Treatment B"), nlas=3,pointwise=FALSE)
```

`analyze.KM.stepp` *Analyze survival data using Kaplan-Meier method*

Description

This method will be deprecated in the future. Please use the constructor function stepp.KM to create a STEPP Kaplan-Meier model for future development.

A method to explore the treatment-covariate interactions in survival data arising from two treatment arms of a clinical trial. The treatment effects are measured using survival functions at a specified time point estimated from the Kaplan-Meier method and the hazard ratio based on observed-minus-expected estimation. A permutation distribution approach to inference is implemented, based on permuting the covariate values within each treatment group. The statistical significance of observed heterogeneity of treatment effects is calculated using permutation tests:

1) for the maximum difference between each subpopulation effect and the overall population treatment effect or supremum based test statistic;
2) for the difference between each subpopulation effect and the overall population treatment effect, which resembles the chi-square statistic.

Usage

```
analyze.KM.stepp(coltrt, coltime, colcens, covar, trts, patspop, minpatspop,
  timest, noperm=2500, ncex = 0.7, legendy = 30, pline = -2.5,
  color = c("red", "black"),
  xlabel = "Subpopulations by Median Covariate",
  ylabel = "?-year Disease-Free Survival",
  tlegend = c("1st Treatment", "2nd Treatment"),
  nlas = 3, pointwise = FALSE)
```

Arguments

`coltrt`	the treatment variable
`coltime`	the time to event variable
`colcens`	the censoring variable
`covar`	the covariate of interest
`trts`	a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment arms, respectively
`patspop`	larger parameter(r2) for subpopulation construction that determines how many patients are in each subpopulation
`minpatspop`	smaller parameter(r1) for subpopulation construction that determines the largest number of patients in common among consecutive subpopulations
`timest`	timepoint to estimate survival
`noperm`	the desired number of permutations; must be 0 or above
`ncex`	optional - specify the size of the text for the sample size annotation, that is, the character expansion factor
`legendy`	optional - the vertical location of the legend according to the units on the y-axis
`pline`	optional - specify the vertical location of the p-value, that is, on which margin line, starting at 0 counting outwards
`color`	optional - a vector containing the line colors for the 1st and 2nd treatment, respectively
`ylabel`	optional - specify the label for the y-axis
`xlabel`	optional - specify the label for the x-axis
`tlegend`	optional - a vector containing the treatment labels, 1st and 2nd treatment, respectively
`nlas`	optional - specify the las parameter (0,1,2,3) to determine the orientation of the sample size annotation
`pointwise`	optional -specify pointwise confidence intervals (pointwise=TRUE), or confidence bands (pointwise=FALSE, default) to be displayed

Details

A statistical method to explore treatment-covariate interactions in survival data arising from two treatment arms of a clinical trial. The method is based on constructing overlapping subpopulations of patients with respect to one covariate of interest, and in observing the pattern of the treatment effects estimated across subpopulations. A plot of these treatment effects is called STEPP, or Subpopulation Treatment Effect Pattern Plot. STEPP uses the permutation distribution based approach for inference.

One can explore the window parameters without invoking the permutation analysis by setting noperm to 0. In that case, pvalue and the covarance matrix will not be available.

Value

analyze.KM.stepp generates a Subpopulation Treatment Effect Pattern Plot (STEPP) displaying the p-value from the test for interaction. Descriptive summaries of the dataset and the estimated variance-covariance matrix are returned through the steppes object. See documentation on steppes object for details on how you can use it.

Warning

This function together with other old functions will be depreciated in the future. A new set of S4 classes are implemented to replace old interfaces. Please use them for future development.

A few tips to keep in mind:

The variables coltrt, coltime, colcens, and covar must be numeric. No formatting allowed.

If you receive the error "Error in solve.default(sigma): system is computationally singular; reciprocal condition number = 0" then we recommend changing the seed by re-running analyze.KM.stepp. If this error persists after several runs, then the program cannot provide reliable results. Please try modifying your choices of the two parameters minpatspop(r1) and patspop(r2) that define the subpopulation.

The number of permutations specified in noperm, the sample size, and the number of subpopulations generated will affect how long analyze.KM.stepp takes to execute. The results are stable if 2500 or more permutations are specified. Furthermore, varying the number of subpopulations will affect inference.

The time point selected to estimate survival in timest must be in the same units (e.g., months) as the coltime variable.

The order of the treatments in the vector trts must be in the same order in the vector tlegend.

Note

STEPP is an exploratory tool, with graphical features that make it easy for clinicians to interpret the results of the analysis. The method provides an opportunity to detect interactions other than those that may be apparent for example by using Cox models. Positive results should prompt the need for confirmation from other datasets investigating similar treatment comparisons. It should also be clear that STEPP is not meant to estimate specific cutpoints in the range of values of the covariate of interest, but rather to provide some indication on ranges of values where treatment effect might have a particular behavior.

STEPP considers the case in which the subpopulations are constructed according to a sliding window pattern. The larger parameter (patspop) determines how many patients are in each subpopulation, and the smaller parameter (minpatspop) determines the largest number of patients in common among consecutive subpopulations. A minimum of 80-100 patients should probably be in each subpopulation, but that is not strictly necessary. The difference (patspop-minpatspop) is the approximate number of patients replaced between any two subsequent subpopulations, and can be used to determine the number of subpopulations once patspop is fixed. The choice of the values of the parameters patspop and minpatspop to be used does change the appearance of the plot and the corresponding p-value. It is probably reasonable to experiment with a few combinations to ensure that the significance (or lack of significance) is stable with respect to that choice.

For best results, consider implementing 2500 permutations of the covariate (vector of subpopulations) to obtain a rich distribution in which to draw inference.

Author(s)

Wai-ki Yip, David Zahrieh, Marco Bonetti, Bernard Cole, Ann Lazar, Richard Gelber

References

Bonetti M, Gelber RD. Patterns of treatment effects in subsets of patients in clinical trials. Biostatistics 2004; 5(3):465-481.

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

See Also

`stwin`, `stsubpop`, `stmodelKM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.test`, `estimate`, `generate`

Old functions to be deprecated: stepp, stepp_summary, stepp_print, stepp_plot, and analyze.CumInc.stepp.

Examples

```
#GENERATE TREATMENT VARIABLE:
N <- 1000
Txassign <- sample(c(1,2), N, replace=TRUE, prob=c(1/2, 1/2))
n1 <- length(Txassign[Txassign==1])
n2 <- N - n1
#GENERATE A COVARIATE:
covariate <- rnorm(N, 55, 7)
#GENERATE SURVIVAL AND CENSORING VARIABLES ASSUMING A TREATMENT COVARIATE INTERACTION:
Entry <- sort( runif(N, 0, 5) )
SurvT1 <- .5
beta0 <-  -65 / 75
beta1 <- 2 / 75
Surv <- rep(0, N)
lambda1 <- -log(SurvT1) / 4
Surv[Txassign==1] <- rexp(n1, lambda1)
Surv[Txassign==2] <- rexp(n2, (lambda1*(beta0+beta1*covariate[Txassign==2])))
EventTimes <- rep(0, N)
EventTimes <- Entry + Surv
censor <- rep(0, N)
time <- rep(0,N)
for ( i in 1:N )
  {
   censor[i] <- ifelse( EventTimes[i] <= 7, 1, 0 )
   time[i] <- ifelse( EventTimes[i] < 7, Surv[i], 7 - Entry[i] )
  }

#CALL analyze.KM.stepp to analyze the data
# Warning: In this example, the permutations have been set to 0 to allow
# the stepp function to finish in a short amount of time.  IT IS RECOMMEND
# TO USE AT LEAST 2500 PERMUTATIONS TO PROVIDE STABLE RESULTS.
output <- analyze.KM.stepp ( coltrt=Txassign, coltime=time,
  colcens=censor, covar=covariate, trts=c(1,2), patspop=300,
  minpatspop=200, timest=4, noperm=0, ncex=0.70, legendy=30,
  pline=-2.5, color=c("red", "black"),
  xlabel="Subpopulations by Median Age", ylabel="4-year Disease-Free Survival",
  tlegend=c("Treatment A", "Treatment B"), nlas=3, pointwise=FALSE)
```

aspirin — *The aspirin data set.*

Description

The "aspirin" data set was obtained with permission from the Polyp Prevention Study Group. The data are from a randomized clinical trial comparing two dosages of daily aspirin (81 mg or 325 mg/day) versus placebo for the prevention of colorectal adenomas among people having a prior history of these lesions. The study randomized a total of 1,121 participants to the three treatment groups. After an approximately three-year treatment period, 1,084 participants underwent follow-up colonoscopy. The primary endpoint of the trial was the occurrence of one or more large-bowel adenomas. Secondary endpoints included the occurrence of any advanced lesion (i.e., adenomas having high-risk characteristics or invasive cancer).

Usage

```
data(aspirin)
```

Format

The data set has the following columns: DOSE - Randomized aspirin dose (0=placebo); AGE - Age of the participant in years at baseline; G - Gender of the participant (M or F) AD - Occurrence of any adenoma at follow-up (0=no, 1=yes, .=missing); AL - Occurrence of any advanced lesion at follow-up(0=no, 1=yes, .=missing)

Source

The Polyp Prevention Study Group is acknowledged for granting permission to use the data from the Aspirin/Folate Polyp Prevention study.

References

Baron J, Cole BF, Sandler RS, et al (2003), "A Randomized Trial of Aspirin to prevent Colorectal Adenomas." n engl j med 2003;234:891-9.

Examples

```
data(aspirin)

# remove cases with missing data
aspirinc <- aspirin[complete.cases(aspirin),]

# make a subset of patients with placebo and 81 mg
attach(aspirinc)
subset1  <- DOSE == 0 | DOSE == 81
```

```
aspirin1 <- aspirinc[subset1,]
detach(aspirinc)

# set up treatment assignment
trtA     <- rep(0, dim(aspirin1)[1])
trtA[aspirin1[,"DOSE"] == 81] <- 1

# STEPP analysis A: placebo vs. 81 mg aspirin
attach(aspirin1)
inc_win     <- stepp.win(type="sliding", r1=30, r2=100)
inc_sp      <- stepp.subpop(swin=inc_win, cov=AGE)

ADorLE      <- as.numeric(AD==1 | AL==1)
modelA      <- stepp.GLM(coltrt=trtA, trts=c(0,1), colY=ADorLE, glm="binomial")
# Warning: In this example, the permutations have been set to 50 to allow the function
# to finish in a short amount of time.  IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO
# PROVIDE STABLE RESULTS.
steppGLMA   <- stepp.test(inc_sp, modelA, nperm=50)
summary(steppGLMA)
print(steppGLMA)
plot(steppGLMA, ncex=0.70, legendy=30,
pline=-4.5, color=c("red","black"),
xlabel="Subpopulations by Median Age", ylabel="Risk",
tlegend=c("Placebo", "81 mg aspirin"), nlas=3, pointwise=FALSE, noyscale=TRUE, rug=FALSE)

detach(aspirin1)
```

`balance_example` *Sample data to use with the* `balance_patients()` *function.*

Description

A sample data set containing a single column which is used to show the usage of the `balance_patients()` function.

Usage

```
data(balance_example)
```

Format

There is only one numeric column: covar (continuous covariate).

References

Lazar A, Cole B, Bonetti M, Gelber R (2010), "Evaluation of treatment-effect heterogeneity using biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot." J Clin Oncol, 28(29), 4539-44.

See Also

`balance_patients()`

Examples

```
## Not run:
data(balance_example, package = "stepp")
ranger2 <- c(950, 1050)
ranger1 <- c(300, 500)
maxnsubpops <- 50

res_bal <- balance_patients(ranger1, ranger2, maxnsubpops, balance_example$covar,
  plot = TRUE, verbose = TRUE, contour = TRUE, nlevels = 6)

## End(Not run)
```

`balance_patients` *Utility function for determining the optimal values for generating the subpopulations.*

Description

Utility function for determining the optimal values of the number of subpopulations and the corresponding r1 and r2 values for creating subpopulations with the sliding window approach. The optimal values are those that make the subpopulations more balanced by minimizing the variance of the subpopulation sizes.

Usage

```
balance_patients(range.r1, range.r2, maxnsubpops, covar, verbose = FALSE,
  plot = FALSE, contour = FALSE, nlevels = 5, showstatus = TRUE)
```

Arguments

`range.r1`	numeric vector with two elements providing the range of values for the r1 parameter
`range.r2`	numeric vector with two elements providing the range of values for the r2 parameter
`maxnsubpops`	length-one numeric vector providing the maximum number of subpopulations to consider
`covar`	numeric vector containing the covariate values to use for generating the subpopulations
`verbose`	length-one logical vector; if `TRUE` prints a summary of the results in the console
`plot`	length-one logical vector; if `TRUE` produces a diagram showing the results of the calculations

`contour`	length-one logical vector; if `TRUE` adds to the plot the variance contour lines for each subpopulation number
`nlevels`	length-one numeric vector providing the number of contour lines to plot
`showstatus`	length-one logical vector; if `TRUE` displays a bar showing the progress of the calculations; default is TRUE

Value

The `balance_patients()` function returns a list with the following items:

`r1_best`	length-one numeric vector with overall best value of the r1 parameter
`r2_best`	length-one numeric vector with overall best value of the r2 parameter
`var_best`	length-one numeric vector with overall minimum value of the sizes variance
`nsubpops_best`	length-one numeric vector with overall best value for the number of subpopulations
`all_res`	numeric matrix with the details of all the calculations

Author(s)

Marco Bonetti, Sergio Venturini

References

Bonetti M, Gelber RD. Patterns of treatment effects in subsets of patients in clinical trials. Biostatistics 2004; 5(3):465-481.

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

Lazar AA, Cole BF, Bonetti M, Gelber RD. Evaluation of treatment-effect heterogeneity using biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot. Journal of Clinical Oncology, 2010; 28(29): 4539-4544.

See Also

`stwin`, `stsubpop`, `stepp.win`, `stepp.subpop`, `stepp.KM`

Examples

```
## Not run:
data(balance_example, package = "stepp")
ranger2 <- c(950, 1050)
ranger1 <- c(300, 500)
maxnsubpops <- 50

res_bal <- balance_patients(ranger1, ranger2, maxnsubpops, balance_example$covar,
  plot = TRUE, verbose = TRUE, contour = TRUE, nlevels = 6)

## End(Not run)
```

bigCI *The BIG 1-98 trial dataset for cumulative incidence STEPP.*

Description

This data set contains 2,685 patients in the Breast International Group (BIG) 1-98 randomized clinical trial. The BIG 1-98 is a Phase III clinical trial of 8,010 post menopausal women with hormone-receptor-positive early invasive breast cancer who were randomly assigned adjuvant therapy of letrozole or tamoxifen. Patterns of treatment effects for varying levels of the biomarker Ki-67 labeling index, a measure of cell proliferation, were analyzed using STEPP. The STEPP analysis showed that letrozole was more effective than tamoxifen for patients with tumors expressing the highest levels of the Ki-67 labeling index. The two treatment arms are letrozole and tamoxifen.

Usage

```
data(bigCI)
```

Format

There are four columns of numeric values: trt (treatment group), time (time to event), event (competing event types), and ki67 (continuous measurement of biomarker Ki-67).

Source

The Breast International Group (BIG) 1-98 Steering Committee and the International Breast Cancer Study Group (IBCSG) are acknowledged for permission to use the data from the BIG 1-98 trial.

References

Lazar A, Cole B, Bonetti M, Gelber R (2010), "Evaluation of treatment-effect heterogeneity using biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot." J Clin Oncol, 28(29), 4539-44.

Viale G et al (2008), "Prognostic and predictive value of cnetrally reviewed Ki-67 labeling index in postmenopausal women with endocrine-responsive breast cancer: results from Breast International Group Trial 1-98 comparing adjuvant tamoxifen and letrozole." J Clin Oncol, 28(34), 5569-75.

Examples

```
data(bigCI)

rxgroup <- bigCI$trt
time    <- bigCI$time
evt     <- bigCI$event
cov     <- bigCI$ki67

# analyze using Cumulative Incidence method with
# sliding window size of 150 patients and a maximum of 50 patients in common
#
```

```
swin    <- new("stwin", type="sliding", r1=50, r2=150) # create a sliding window
subp    <- new("stsubpop")                              # create subpopulation object
subp    <- generate(subp, win=swin, covariate=cov) # generate the subpopulations
summary(subp)    # summary of the subpopulations

# create a stepp model using Cumulative Incidences to analyze the data
#
smodel <- new("stmodelCI", coltrt=rxgroup, trts=c(1,2), coltime=time, coltype=evt, timePoint=4)

statCI  <- new("steppes")   # create a test object based on subpopulation and window
statCI  <- estimate(statCI, subp, smodel) # estimate the subpopulation results
# Warning: In this example, the permutations have been set to 0 to allow the function
# to finish in a short amount of time.  IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO
# PROVIDE STABLE RESULTS.
statCI  <- test(statCI, nperm=0)       # permutation test with 0 iterations

print(statCI)   # print the estimates and test statistics
plot(statCI, ncex=0.65, legendy=30, pline=-15.5, color=c("blue","gold"),
        pointwise=FALSE,
        xlabel="Median Ki-67 LI in Subpopulation (% immunoreactivity)",
        ylabel="4-year Cumulative Incidence",
        tlegend=c("Letrozole", "Tamoxifen"), nlas=3)
```

bigKM *The BIG 1-98 trial dataset for Kaplan-Meier STEPP.*

Description

This data set contains 2,685 patients in the Breast International Group (BIG) 1-98 randomized clinical trial. The BIG 1-98 is a Phase III clinical trial of 8,010 post menopausal women with hormone-receptor-positive early invasive breast cancer who were randomly assigned adjuvant therapy of letrozole or tamoxifen. Patterns of treatment effects for varying levels of the biomarker Ki-67 labeling index, a measure of cell proliferation, were analyzed using STEPP. The STEPP analysis showed that letrozole was more effective than tamoxifen for patients with tumors expressing the highest levels of the Ki-67 labeling index. The two treatment arms are letrozole and tamoxifen.

Usage

```
data(bigKM)
```

Format

There are four columns of numeric values: trt (treatment group), time (time to event), event (competing event types), and ki67 (continuous measurement of biomarker Ki-67).

Source

The Breast International Group (BIG) 1-98 Steering Committee and the International Breast Cancer Study Group (IBCSG) are acknowledged for permission to use the data from the BIG 1-98 trial.

References

Lazar A, Cole B, Bonetti M, Gelber R (2010), "Evaluation of treatment-effect heterogeneity using biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot." J Clin Oncol, 28(29), 4539-44.

Viale G et al (2008), "Prognostic and predictive value of cnetrally reviewed Ki-67 labeling index in postmenopausal women with endocrine-responsive breast cancer: results from Breast International Group Trial 1-98 comparing adjuvant tamoxifen and letrozole." J Clin Oncol, 28(34), 5569-75.

Examples

```
data(bigKM)

rxgroup <- bigKM$trt
time    <- bigKM$time
evt     <- bigKM$event
cov     <- bigKM$ki67

# analyze using Cumulative Incidence method with
# sliding window size of 150 patients and a maximum of 50 patients in common
#
swin    <- new("stwin", type="sliding", r1=50, r2=150) # create a sliding window
subp    <- new("stsubpop")                             # create subpopulation object
subp    <- generate(subp, win=swin, covariate=cov) # generate the subpopulations
summary(subp)    # summary of the subpopulations

# create a stepp model using Kaplan Meier Method to analyze the data
#
smodel  <- new("stmodelKM", coltrt=rxgroup, trts=c(1,2), survTime=time, censor=evt, timePoint=4)

statKM  <- new("steppes")   # create a test object based on subpopulation and window
statKM  <- estimate(statKM, subp, smodel) # estimate the subpopulation results
# Warning: In this example, the permutations have been set to 0 to allow the function
# to finish in a short amount of time.  IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO
# PROVIDE STABLE RESULTS.
statKM  <- test(statKM, nperm=0)       # permutation test with 0 iterations

print(statKM)   # print the estimates and test statistics
plot(statKM, ncex=0.65, legendy=30, pline=-15.5, color=c("blue","gold"),
        pointwise=FALSE,
        xlabel="Median Ki-67 LI in Subpopulation (% immunoreactivity)",
        ylabel="4-year Disease Free Survival",
        tlegend=c("Letrozole", "Tamoxifen"), nlas=3)
```

`estimate` *The standard generic function for all estimate methods*

Description

The default generic function for estimate methods for all stepp models classes and the steppes class. For detail, please refer to the documentation in the estimate method in the S4 class: steppes.

Author(s)

Wai-ki Yip

See Also

stwin, stsubpop, stmodelKM, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.GLM, stepp.test, generate

---

gen.tailwin *Utility function to generate tail-oriented window*

Description

Utility function to generate tail-oriented windows inputs given the approximate number of subpopulations desired.

Usage

```
gen.tailwin(covariate, nsub, dir="LE")
```

Arguments

covariate	covariate values
nsub	number of tail-oriented subpopulations to be generated
dir	"LE" (default) or "GE" - subpopulations with covariate values less than or equal/greater than or equal to the generated values

Details

Use this together with the constructor, stepp.win, to generate tail-oriented windows.

Value

It returns a list with fields: $v - vector of covariate values to be used in the constructor stepp.win. $np - vector of subpopulation size associate with each tail-oriented window defined by $v.

Author(s)

Wai-ki Yip

See Also

stwin, stepp.win

Examples

```
data(bigKM)

rxgroup <- bigKM$trt
time    <- bigKM$time
evt     <- bigKM$event
cov     <- bigKM$ki67

# analyze using Kaplan-Meier method with tail-oriented window
#
nsubpop_tmp <- 10
win_tmp <- gen.tailwin(cov, nsub = nsubpop_tmp, dir = "LE")
nsubpop <- length(win_tmp$v)
# create a tail-oriented window
swin <- new("stwin", type = "tail-oriented", r1 = win_tmp$v, r2 = rep(min(cov), nsubpop))
subp <- new("stsubpop")                              # create subpopulation object
subp <- generate(subp, win = swin, covariate = cov) # generate the subpopulations
summary(subp)                              # summary of the subpopulations

# create a stepp model using Kaplan Meier Method to analyze the data
#
smodel  <- new("stmodelKM", coltrt=rxgroup, trts=c(1,2), survTime=time, censor=evt, timePoint=4)

statKM  <- new("steppes")     # create a test object based on subpopulation and window
statKM  <- estimate(statKM, subp, smodel) # estimate the subpopulation results
# Warning: IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO  PROVIDE STABLE RESULTS.
statKM <- test(statKM, nperm = 10)       # permutation test with 10 iterations

print(statKM)         # print the estimates and test statistics
plot(statKM, ncex=0.65, legendy=30, pline=-15.5, color=c("blue","gold"),
     pointwise=FALSE,
     xlabel="Median Ki-67 LI in Subpopulation (% immunoreactivity)",
     ylabel="4-year Disease Free Survival",
     tlegend=c("Letrozole", "Tamoxifen"), nlas=3)
```

`generate` *The standard generic function for the generate method in stsubpop class*

Description

The default generic function for the generate method for stsubpop class.

For detail, please refer to the documentation in the generate method in the S4 class: stsubpop.

Author(s)

Wai-ki Yip

See Also

stwin, stsubpop, stmodelKM, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.GLM, stepp.test, estimate

---

simdataKM *Simulated data for Kaplan-Meier STEPP analysis.*

Description

Simulated data to use in a Kaplan-Meier STEPP analysis.

Usage

```
data(simdataKM)
```

Format

A data.frame object with columns:

time time-to-event data

censor censoring indicator

trt treatment indicator

covar covariate values

Examples

```
### the following code generates the data in the object ###

set.seed(101)
n <- 1000

# generate the treatment indicator
Txassign <- sample(c(1, 2), n, replace = TRUE, prob = rep(.5, 2))
n1 <- sum(Txassign == 1)
n2 <- n - n1

# generate the covariate values
mean_cov <- 55
sd_cov <- 7
covariate <- rnorm(n, mean = mean_cov, sd = sd_cov)

# generate the survival times and censoring indicator assuming a treatment-covariate interaction
Entry <- sort(runif(n, 0, 5))
SurvT1 <- .5
beta0 <- -65/75
beta1 <- 2/75
Surv <- rep(0, n)
lambda1 <- -log(SurvT1)/4
```

```
Surv[Txassign == 1] <- rexp(n1, lambda1)
Surv[Txassign == 2] <- rexp(n2, (lambda1*(beta0 + beta1*covariate[Txassign == 2])))
EventTimes <- rep(0, n)
EventTimes <- Entry + Surv
censor <- time <- rep(0, n)
for (i in 1:n) {
  censor[i] <- ifelse(EventTimes[i] <= 7, 1, 0)
  time[i] <- ifelse(EventTimes[i] < 7, Surv[i], 7 - Entry[i])
}
simdataKM <- data.frame(time = time, censor = censor, trt = Txassign, covar = covariate)

# overall survival analysis
simdataKM_surv <- Surv(simdataKM$time, simdataKM$censor)
simdataKM_fit <- survfit(simdataKM_surv ~ trt, data = simdataKM)
plot(simdataKM_fit, lty = 2:3, lwd = rep(2, 2), col = 2:3)
legend("topright", c("Treatment 1", "Treatment 2"), lty = 2:3, lwd = rep(2, 2), col = 2:3)
title("Kaplan-Meier Curves\nfor simulated data")
survdiff(simdataKM_surv ~ trt, data = simdataKM, rho = 0)
```

---

stepp *Analyze survival or competing risks data*

Description

This method will be deprecated in the future. Please use S4 classes (stmodelCI, stmodelKM) and corresponding constructors (stepp.CI and stepp.KM) for future development.

A method to explore the treatment-effect interactions in either survival or competing risks data arising from two treatment arms of a clinical trial. A permutation distribution approach to inference is implemented, based on permuting covariate values within each treatment group. The statistical significance of observed heterogeneity of treatment effects is calculated using permutation tests:

1) for the maximum difference between each subpopulation effect and the overall population treatment effect or supremum based test statistic;
2) for the difference between each subpopulation effect and the overall population treatment effect, which resembles the chi-square statistic.

Usage

```
stepp(trttype, coltrt, coltime, colcens=0, coltype=0, covar, trts,
      patspop, minpatspop, timest, noperm)
```

Arguments

`trttype`	type of analysis methods to use: "KM" for Kaplan-Maier and "CI" for Cumulative Incidence
`coltrt`	the treatment variable
`coltime`	the time to event variable

`colcens`	the censoring variable. If specified, the KM method will be used
`coltype`	variable with distinct codes for different causes of failure where coltype=0 for censored observations; coltype=1 for event of interest; coltype=2 for other causes of failure. If specified, the cumulative incidence method will be used.
`covar`	the covariate of interest
`trts`	a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment arms, respectively
`patspop`	larger parameter(r2) for subpopulation construction that determines how many patients are in each subpopulation
`minpatspop`	smaller parameter(r1) for subpopulation construction that determines the largest number of patients in common among consecutive subpopulations
`timest`	timepoint to estimate survival
`noperm`	the desired number of permutations; must be 0 or above

Details

A statistical method to explore treatment by covariate interactions in survival or competing risks data arising from two treatment arms of a clinical trial. The method is based on constructing overlapping subpopulations of patients with respect to a covariate of interest, and in observing the pattern of the treatment effects estimated across subpopulations. A plot of these treatment effects is called STEPP, or Subpopulation Treatment Effect Pattern Plot. STEPP uses the permutation distribution based approach for inference.

One can explore the window parameters without invoking the permutation analysis by setting noperm to 0. In that case, pvalue and the covarance matrix will not be available.

We acknowledge Robert J. Gray for permitting us to use the cmprsk package.

Value

stepp does all the computation and anlaysis but does not generate any Subpopulation Treatment Effect Pattern Plot (STEPP). A steppes object is returned and the user can use stepp_summary, stepp_print, or stepp_plot to display the information and to generate the plots for further analysis.

Warning

This function together with other old functions will be depreciated in the future. A new set of S4 classes are implemented to replace old interfaces. Please use them for future development.

A few tips to keep in mind:

The variables coltrt, coltime, coltype, and covar must be numeric. No formatting allowed.

If you receive the error "Error in solve.default(sigma): system is computationally singular; reciprocal condition number = 0" then we recommend changing the seed by re-running stepp. If this error persists after several runs, then the program cannot provide reliable results. Please try modifying your choices of the two parameters minpatspop(r1) and patspop(r2) that define the subpopulation.

The number of permutations specified in noperm, the sample size, and the number of subpopulations generated will affect how long stepp takes to execute. The results are stable if 2500 or more permutations are specified. Furthermore, varying the number of subpopulations will affect inference.

The time point selected to estimate survival in timest must be in the same units (e.g., months) as the coltime variable.

The order of the treatments in the vector trts must be in the same order in the vector tlegend.

Note

STEPP is an exploratory tool, with graphical features that make it easy for clinicians to interpret the results of the analysis. Positive results should prompt the need for confirmation from other datasets investigating similar treatment comparisons. It should also be clear that STEPP is not meant to estimate specific cutpoints in the range of values of the covariate of interest, but rather to provide some indication on ranges of values where treatment effect might have a particular behavior.

STEPP considers the case in which the subpopulations are constructed according to a sliding window pattern. The larger parameter (patspop) determines how many patients are in each subpopulation, and the smaller parameter (minpatspop) determines the largest number of patients in common among consecutive subpopulations. A minimum of 80-100 patients should be in each subpopulation, but that is not strictly necessary. The difference (patspop-minpatspop) is the approximate number of patients replaced between any two subsequent subpopulations, and can be used to determine the number of subpopulations once patspop is fixed. The choice of the values of the parameters patspop and minpatspop to be used does change the appearance of the plot and the corresponding p-value. It is probably reasonable to experiment with a few combinations to ensure that the significance (or lack of significance) is stable with respect to that choice.

For best results, consider implementing 2500 permutations of the covariate (vector of subpopulations) to obtain a rich distribution in which to draw inference.

Author(s)

Ann Lazar, Wai-ki Yip, David Zahrieh, Bernard Cole, Marco Bonetti, Richard Gelber

References

Bonetti M, Gelber RD. Patterns of treatment effects in subsets of patients in clinical trials. Biostatistics 2004; 5(3):465-481.

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

Lazar AA, Cole BF, Bonetti M, Gelber RD. Evaluation of treatment-effect heterogeneity using biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot. Journal of Clinical Oncology, 2010; 28(29): 4539-4544.

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.test`, `estimate`, `generate`,

Old functions to be deprecated: stepp_summary, stepp_print, stepp_plot, analyze.KM.stepp and analyze.CumInc.stepp.

Examples

```
#GENERATE TREATMENT VARIABLE:
n <- 1000 # set the sample size
mu <- 0 # set the mean and sd of the covariate
sigma <- 1

beta0 <- log(-log(0.5)) # set the intercept for the log hazard
beta1 <- -0.2 # set the slope on the covariate
beta2 <- 0.5 # set the slope on the treatment indicator
beta3 <- 0.7 # set the slope on the interaction

prob2 <- 0.2 # set the proportion type 2 events
cprob <- 0.3 # set the proportion censored

set.seed(7775432)  # set the random number seed
covariate <- rnorm(n,mean=mu,sd=sigma) # generate the covariate values
Txassign <- rbinom(n,1,0.5) # generate the treatment indicator
x3 <- covariate*Txassign # compute interaction term
# compute the hazard for type 1 event
lambda1 <- exp(beta0+beta1*covariate+beta2*Txassign+beta3*x3)
lambda2 <- prob2*lambda1/(1-prob2) # compute the hazard for the type 2 event
# compute the hazard for censoring time
lambda0 <- cprob*(lambda1+lambda2)/(1-cprob)
t1 <- rexp(n,rate=lambda1) # generate the survival time for type 1 event
t2 <- rexp(n,rate=lambda2) # generate the survival time for type 2 event
t0 <- rexp(n,rate=lambda0) # generate the censoring time
time <- pmin(t0,t1,t2) # compute the observed survival time
type <- rep(0,n)
type[(t1 < t0)&(t1 < t2)] <- 1
type[(t2 < t0)&(t2 < t1)] <- 2

# Call stepp to analyze the data
# Warning: In this example, the permutations have been set to 0 to allow the stepp function
# to finish in a short amount of time.  IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO
# PROVIDE STABLE RESULTS.
steppCI <- stepp("CI", coltrt=Txassign, trts=c(0,1), coltime=time, coltype=type,
  covar=covariate, patspop=300, minpatspop=200, timest=1.0, noperm=0)

stepp_summary(steppCI)
stepp_print(steppCI)
stepp_plot(steppCI,
  ncex=0.70,legendy=30,pline=-2.5,color=c("red", "black"),
  xlabel="Subpopulations by Median Age",ylabel="4-year Cancer Relapse",
  tlegend=c("Treatment A", "Treatment B"), nlas=3,pointwise=FALSE)
```

`stepp.CI` *The constructor to create the stmodelCI object*

Description

This is the constructor function for the stmodelCI object. This object sets up the data with a stepp model using competing risks method for analysis. (CI stands for Cumulative Incidence.)

The model explores the treatment-effect interactions in competing risks data arising from two or more treatment arms of a clinical trial. A permutation distribution approach to inference is implemented that permutes covariate values within a treatment group. The statistical significance of observed heterogeneity of treatment effects is calculated using permutation tests:

1) for the maximum difference between each subpopulation effect and the overall population treatment effect or supremum based test statistic;
2) for the difference between each subpopulation effect and the overall population treatment effect, which resembles the chi-square statistic.

Usage

```
stepp.CI(coltrt, coltime, coltype, trts, timePoint)
```

Arguments

`coltrt`	the treatment variable
`coltime`	the time to event variable
`coltype`	variable with distinct codes for different causes of failure where coltype=0 for censored observations; coltype=1 for event of interest; coltype=2 for other causes of failure
`trts`	a vector containing the codes for the 2 treatment groups, 1st and 2nd treatment groups, respectively
`timePoint`	timepoint to estimate survival

Value

It returns the stmodelCI object.

Author(s)

Wai-Ki Yip

References

Bonetti M, Gelber RD. Patterns of treatment effects in subsets of patients in clinical trials. Biostatistics 2004; 5(3):465-481.

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

Lazar AA, Cole BF, Bonetti M, Gelber RD. Evaluation of treatment-effect heterogeneity usiing biomarkers measured on a continuous scale: subpopulation treatment effect pattern plot. Journal of Clinical Oncology, 2010; 28(29): 4539-4544.

See Also

stwin, stsubpop, stmodelKM, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.GLM, stepp.test, estimate, generate

Examples

```
##
n <- 1000 # set the sample size
mu <- 0 # set the mean and sd of the covariate
sigma <- 1

beta0 <- log(-log(0.5)) # set the intercept for the log hazard
beta1 <- -0.2 # set the slope on the covariate
beta2 <- 0.5 # set the slope on the treatment indicator
beta3 <- 0.7 # set the slope on the interaction

prob2 <- 0.2 # set the proportion type 2 events
cprob <- 0.3 # set the proportion censored

set.seed(7775432)  # set the random number seed
covariate <- rnorm(n,mean=mu,sd=sigma) # generate the covariate values
Txassign <- rbinom(n,1,0.5) # generate the treatment indicator
x3 <- covariate*Txassign # compute interaction term
# compute the hazard for type 1 event
lambda1 <- exp(beta0+beta1*covariate+beta2*Txassign+beta3*x3)
lambda2 <- prob2*lambda1/(1-prob2) # compute the hazard for the type 2 event
# compute the hazard for censoring time
lambda0 <- cprob*(lambda1+lambda2)/(1-cprob)
t1 <- rexp(n,rate=lambda1) # generate the survival time for type 1 event
t2 <- rexp(n,rate=lambda2) # generate the survival time for type 2 event
t0 <- rexp(n,rate=lambda0) # generate the censoring time
time <- pmin(t0,t1,t2) # compute the observed survival time
type <- rep(0,n)
type[(t1 < t0)&(t1 < t2)] <- 1
type[(t2 < t0)&(t2 < t1)] <- 2

# create the stepp model object to analyze the data using Cumulative Incidence approach
x <- stepp.CI(coltrt=Txassign, trts=c(0,1), coltime=time, coltype=type, timePoint=1.0)
```

`stepp.edge` *The method performs an edge analysis on the STEPP GLM model estimate objects.*

Description

Perform an edge analysis on the STEPP GLM model estimate objects.

Usage

```
stepp.edge(est, criteria, j=2, boot=0, seed=17, showstatus=TRUE, debug=0)
```

Arguments

`est`	a STEPP estimate object.
`criteria`	criteria to be used to identify the cut point; abs or rel scale and by how much.
`j`	number of treatment, default to 2
`boot`	perform a bootstrap analysis, default is none (0).
`seed`	seed used for bootstrap, default is 17.
`showstatus`	show the status of bootstrap, default is TRUE.
`debug`	internal debug flag, default is 0

Details

The criteria argument is a list with three elements:

trtid - the treatment id;

scale - "A" for absolute scale or "R" for relative scale;

threshold - amount in either absolute or relative scale that would consider a jump.

e.g. crit <- list(trtid=1, scale="A", threshold=-0.03)

Only support for STEPP GLM models for now. Bootstrap is computational time intensive but it will provide you with a quantification of the variability of the edge identified.

Value

It returns the result of the identified edge with the STEPP subpopulations.

Author(s)

Wai-ki Yip

See Also

`stwin`

`stepp.GLM` *The constructor to create the stmodelGLM object*

Description

This is the constructor function for stmodelGLM object. Three kinds of GLMs are supported at this time: gaussian with identity link, binomial with logit link and Poisson with log link.

Usage

```
stepp.GLM(coltrt, trts, colY, MM, glm, link)
```

Arguments

`coltrt`	the treatment variable
`trts`	a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment groups, respectively
`colY`	a vector containing the outcome
`MM`	a model matrix for additional covariates; default is NULL no logical values or factors allowed these values need to be converted to numeric with some encoding scheme
`glm`	the glm to be used for analysis: "gaussian", "binomial", "poisson"
`link`	the link function; reserved for future use

Value

It returns a stmodelGLM object.

Author(s)

Wai-Ki Yip

References

Yip WK, Bonetti M, Cole BF, Barcella W, Wang XV, Lazar A and Gelber R (2016), "STEPP-Subpopulation Analysis for Continuous, Binary and Count Outcomes", Clinical Trials 2016 August ; 13(4): 382-290, doi:10.1177/1740774516643297.

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.test`, `estimate`, `generate`

Examples

```
#
# see the example for the aspirin data for an example for a binomial glm model
#
```

`stepp.KM` *The constructor to create the stmodelKM object*

Description

This is the constructor function for the stmodelKM object. This object sets up the data with a stepp model using the Kaplan-Meier method for analysis.

The model explores the treatment-covariate interactions in survival data arising from two treatment arms of a clinical trial. The treatment effects are measured using survival functions at a specified time point estimated from the Kaplan-Meier method and the hazard ratio based on observed-minus-expected estimation. A permutation distribution approach to inference is implemented, based on permuting the covariate values within each treatment group. The statistical significance of observed heterogeneity of treatment effects is calculated using permutation tests:

1) for the maximum difference between each subpopulation effect and the overall population treatment effect or supremum based test statistic;
2) for the difference between each subpopulation effect and the overall population treatment effect, which resembles the chi-square statistic.

Usage

```
stepp.KM(coltrt, survTime, censor, trts, timePoint)
```

Arguments

`coltrt`	the treatment variable
`survTime`	the time to event variable
`censor`	the censor variable
`trts`	a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment groups, respectively
`timePoint`	timepoint to estimate survival

Value

It returns the stmodelKM object.

Author(s)

Wai-Ki Yip

References

Bonetti M, Gelber RD. Patterns of treatment effects in subsets of patients in clinical trials. Biostatistics 2004; 5(3):465-481.

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

See Also

stwin, stsubpop, stmodelKM, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.CI, stepp.GLM, stepp.test, estimate, generate

Examples

```
#GENERATE TREATMENT VARIABLE:
N <- 1000
Txassign <- sample(c(1,2), N, replace=TRUE, prob=c(1/2, 1/2))
n1 <- length(Txassign[Txassign==1])
n2 <- N - n1
#GENERATE A COVARIATE:
covariate <- rnorm(N, 55, 7)
#GENERATE SURVIVAL AND CENSORING VARIABLES ASSUMING A TREATMENT COVARIATE INTERACTION:
Entry <- sort( runif(N, 0, 5) )
SurvT1 <- .5
beta0 <-  -65 / 75
beta1 <- 2 / 75
Surv <- rep(0, N)
lambda1 <- -log(SurvT1) / 4
Surv[Txassign==1] <- rexp(n1, lambda1)
Surv[Txassign==2] <- rexp(n2, (lambda1*(beta0+beta1*covariate[Txassign==2])))
EventTimes <- rep(0, N)
EventTimes <- Entry + Surv
censor <- rep(0, N)
time <- rep(0,N)
for ( i in 1:N )
   {
     censor[i] <- ifelse( EventTimes[i] <= 7, 1, 0 )
     time[i] <- ifelse( EventTimes[i] < 7, Surv[i], 7 - Entry[i] )
   }

modKM <- stepp.KM( coltrt=Txassign, survTime=time, censor=censor, trts=c(1,2), timePoint=4)
```

`stepp.rnote` *The method to print the release note for STEPP.*

Description

This method will print the current stepp version and the corresponding release note.

Usage

```
  stepp.rnote()
```

Author(s)

Wai-ki Yip, Marco Bonetti, Bernard Cole, Ann Lazar, Victoria Xin Wang, Richard Gelber

Examples

```
stepp.rnote()
```

`stepp.subpop` *The constructor to create the stsubpop object and generate the subpopulations based on the specified stepp window and covariate of interest*

Description

This is the constructor function to create a stepp subpopulation object. In addition, it will also generate the stepp subpopulations based on the stepp window and covariate specified.

Usage

```
stepp.subpop(swin, cov, coltype, coltrt, trts, minsubpops)
```

Arguments

`swin`	the stepp window set up for the analysis
`cov`	the covariate of interest
`coltype`	variable providing the event type; required only for event-based windows
`coltrt`	variable providing the treatment indicators; required only for event-based windows
`trts`	vector containing the treatments list; required only for event-based windows
`minsubpops`	length-one numeric vector providing the minimum number of subpopulations to generate; required only for event-based windows

Value

It returns the stsubpop object with subpopulations generated.

Author(s)

Wai-Ki Yip

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
# create a steppp window
win1 <- stepp.win(type="sliding", r1=5,r2=10)

# generate the covariate of interest
Y <- rnorm(100)

# create and generate the stepp subpopulation
sp   <- stepp.subpop(swin=win1, cov=Y)

# event-based windows using the BIG data set
data(bigKM)
rxgroup <- bigKM$trt
time    <- bigKM$time
evt     <- bigKM$event
cov     <- bigKM$ki67

swin_e <- new("stwin", type = "sliding_events", e1 = 10, e2 = 20)
subp_e <- new("stsubpop")
subp_e <- generate(subp_e, win = swin_e, covariate = cov, coltype = evt,
                   coltrt = rxgroup, trts = c(1, 2), minsubpops = 5)
summary(subp_e)
```

`stepp.test` *The constructor to generate a complete steppes object with effect estimates and test statistics*

Description

This is a constructor function for the steppes object. In addition, it estimates all the effects of each subpopulations, performs permutation tests and generates all covariance matrices and statistics.

Usage

```
stepp.test(subpop, model, nperm, showstatus = TRUE)
```

Arguments

`subpop`	the filled stepp subpopulation object
`model`	the model of the data to be used for stepp analysis
`nperm`	number of permutation used in the permutation test
`showstatus`	display the progress bar for the permutation test; default is TRUE

Details

Permutation tests for all the statistics are done (see ref below). For best results, considering using 2500 permutations to obtain a rich distribution from which to draw inference.

Value

It returns a steppes object with all effects estimates, covariance matrices and statistics.

Author(s)

Wai-ki Yip

References

Bonetti M, Zahrieh D, Cole BF, Gelber RD. A small sample study of the STEPP approach to assessing treatment-covariate interactions in survival data. Statistics in Medicine 2009; 28(8):1255-68.

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `estimate`, `generate`

Examples

```
data(bigCI)

rxgroup <- bigCI$trt
time    <- bigCI$time
evt     <- bigCI$event
cov     <- bigCI$ki67

#
# using constructor functions
swin    <- stepp.win(type="sliding", r1=50, r2=150)
subp    <- stepp.subpop(swin=swin, cov=cov)
summary(subp)

smodel  <- stepp.CI(coltrt=rxgroup, trts=c(1,2), coltime=time, coltype=evt, timePoint=4)
# Warning: In this example, the permutations have been set to 0 to allow the function
# to finish in a short amount of time.  IT IS RECOMMEND TO USE AT LEAST 2500 PERMUTATIONS TO
# PROVIDE STABLE RESULTS.
statCI  <- stepp.test(subpop=subp, model=smodel, nperm=0)
```

`stepp.win` *The constructor to create the stepp window object*

Description

This is the constructor function to create a stepp window object.

Usage

```
stepp.win(type = "sliding", r1 = 5, r2 = 20, e1 = NULL, e2 = NULL)
```

Arguments

`type`	type of stepp window; "sliding", "sliding_events" or "tail-oriented"
`r1`	sliding window: minimum number of patients allowed in overlapping windows; tail-oriented window: a vector of maximum covariate values for each subpopulation
`r2`	sliding window: size of subpopulation in each window; tail-oriented window: a vector of minimum covariate values for each subpopulation
`e1`	sliding window: minimum number of events allowed in overlapping windows
`e2`	sliding window: number of events in each subpopulation

Details

This is the functional interface to construct a stepp window object besides using new("stwin", ...). A STEPP window specifies a pattern of subpopulation you would like to explore. There are three kinds of patterns: "sliding window", "event-based sliding window" and "tail-oriented window". These patterns are specified through the following set of values (r1 and r2 for sliding and tail-oriented windows, e1 and e2 for event-based sliding windows):

1. for sliding windows based on the number of patients ("sliding"), r1 is the minimum number of patients allowed in overlapping windows and r2 is the approximate size of subpopulation in each window;

2. for sliding windows based on the number of events ("sliding_events"), e1 is the minimum number of events allowed in overlapping windows and e2 is the approximate size of subpopulation in each window in terms of number of events;

3. for tail-oriented window, r1 is a vector of maximum covariate value for each subpopulation from the minimum value of the entire subpopulation, and r2 is a vector of minimum covariate values for each subpopulation from the maximum value of the window. The utility function, gen.tailwin, can be used to generate these two vectors based on the number of desired subpopulations.

When "sliding_events" is chosen r1 and r2 are set to `NULL`, while when "sliding" or "tail-oriented" are chosen e1 and e2 are set to `NULL`.

These pattern are based on all patients.

Author(s)

Wai-ki Yip

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
# create a stepp window object of type "sliding",
# subpopulation size is 200 and allows only 50 patients
# between overlapping windows
mywin <- stepp.win(type="sliding", r1=50, r2=200)

# print a summary of the stepp window object
summary(mywin)

# create a stepp window object of type "sliding_events",
# (event-based) subpopulation size is 200 and allows
# only 50 events between overlapping windows
mywin <- stepp.win(type="sliding_events", e1=50, e2=200)

# print a summary of the stepp window object
summary(mywin)
```

`steppes-class` *Class* `"steppes"`

Description

This is the stepp object for stepp results. It keeps track of all the estimates, covariance matrices and the test statistics

Value

The new method returns the steppes object.

Objects from the Class

Objects can be created by calls of the form `new("steppes", ...)` or by the constructor function stepp.test.

Slots

`subpop`**:** Object of class `"stsubpop"`
  the filled stepp subpopulation object

`model`**:** Object of class `"stmodel"`
  the model of the data to be used for stepp analysis

`effect`**:** Object of class `"ANY"`
  effect estimates of each subpopulation in absolute and relative scales
  list return from the estimate methods of various stepp models
  see documentation in various stepp models for details on what is in the list

`testresults`**:** Object of class `"ANY"`
  permutation or GEE test results and various covariance matrices
  list return from the test methods of various stepp models
  see documentation in various stepp models for details on what is in the list

`nperm`**:** Object of class `"numeric"`
  number of permutation for the permutation test; default is 2500

Methods

estimate `signature(.Object = "steppes", sp, model)`:
  estimate the effect in absolute and relative scale of the overall and each subpopulation for the specified subpopulation using the specified model.

plot `signature(.Object = "steppes")`:
  generate the three stepp plots

print `signature(.Object = "steppes", estimate=TRUE, cov=TRUE, test=TRUE)`:
  print the estimates, covariance matrices and statistics from the stepp analysis

summary `signature(.Object = "steppes")`:
  produce a summary of the steppes object

test `signature(.Object = "steppes", nperm=100, showstatus=TRUE)`:
  perform the permutation tests and obtain various statistics. Number of permutation is default to 100 and status bar to show.

Note

The plot function for steppes class generates 3 plots at once. If you are using R studio for development, you can only get the last plot as R studio only allows one graphical device. There is a simple workaround. By writing the plots to pdf files in your working directory, you can display them outside of R studio.

Author(s)

Wai-Ki Yip

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`,`stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
showClass("steppes")
```

`stepp_plot` *A function to generate the stepp plots*

Description

This function will be deprecated in the future. Please use S4 classes and its generic function e.g. summary, print and plot for future development.

A method to generate the 3 stepp plots for users to examine the treatment-covariate interactions in various models data arising from two treatment arms of a clinical trial. Three plots are generated:

1. effect estimates of both treatments against the median of each subpopulation
2. effect differences (in absolute scale) between the two treatments against the median of each subpopulation.
3. effect ratios (in relative scale) between the two treatments against the median of each subpopulation.

Usage

```
stepp_plot(x, legendy = 30, pline = -2.5, color = c("red", "black"),
 ylabel= "Specify Timepoint & Endpoint", xlabel="Subpopulations by Median Covariate",
  ncex = 0.7, tlegend=c("Specify 1st Treatment", "Specify 2nd Treatment"),
  nlas = 0, alpha = 0.05, pointwise = FALSE, ci = TRUE, pv = TRUE,
  showss = TRUE, ylimit=c(0,100,-100,100,0,3), which="",
  noyscale=FALSE, at=NA, subplot=FALSE)
```

Arguments

`x`	the steppes object returned from stepp, analyze.KM.stepp or analyze.CumInc.stepp
`legendy`	optional - the vertical location of the legend according to the units on the y-axis; default is 30
`pline`	optional - specify the vertical location of the p-value, that is, on which MARgin line, starting at 0 counting outwards; default is -2.5
`color`	optional - a vector containing the line colors for the 1st and 2nd treatment, respectively; default is "red" and "black"
`ylabel`	optional - specify the label for the y-axis; default is "Specify Timepoint & Endpoint"
`xlabel`	optional - specify the label for the x-axis; default is "Subpopulations by Median Covariate"
`ncex`	optional - specify the size of the text for the sample size annotation, that is, the character expansion factor; default is 0.7
`tlegend`	optional - a vector containing the treatment labels, 1st and 2nd treatment, respectively; default is c("Specify 1st Treatment", "Specify 2nd Treatment"
`nlas`	optional - specify the las paramter (0,1,2,3) to determine the orientation of the sample size annotation; default is 0
`alpha`	optional - specify the significance level; default is 0.05
`pointwise`	optional - specify pointwise confidence intervals (pointwise=TRUE), or confidence bands; default is FALSE
`ci`	optional - specify if you want to display the conf. interval or band; default is TRUE

`pv`	optional - pvalue will be displayed in the plots; FALSE: pvalue will not be displayed in the plots; default is TRUE
`showss`	optional - specify the label for the x-axisoptional - show the sample size; FALSE: sample size will not be shown in the plots; default is TRUE
`ylimit`	optional - pvalue will be displayed in the plots; FALSE: pvalue will not be displayed in the plots; default is TRUE
`which`	optional - if a subset of the plots is required, specify a subset of the numbers 1:3, which is the default
`noyscale`	optional - do not scale to the y-axis; default is FALSE
`at`	optional - specify the horizontal position of the p-value; use together with pline; default is at the minimal value of x
`subplot`	optional - generate a subpopulation distribution subplot on top of the first STEPP plot; default is FALSE (no subplot)

Details

This is the function to generate three stepp plots for user to explore treatment-covariate interactions in various models data arising from two treatment arms of a clinical trial. You can choose to produce a single figure with all the three plots or only a subset of these through the use of the 'which' argument.

Through these plots, the user can examine the differences in effects (either in absolute or relative scale) with respect to each subpopulation.

This is mainly a graphics function. The STEPP analysis is done. Most of the parameters are for graphics control so that one can customize the plots for presentation purposes.

Warning

This function together with other old functions will be depreciated in the future. A new set of S4 classes are implemented to replace old interfaces. Please use them for future development.

Author(s)

Wai-ki Yip, David Zahrieh, Marco Bonetti, Bernard Cole, Ann Lazar, Richard Gelber

See Also

The S4 classes: stwin, stsubpop, stmodelKM, stmodelCI, stmodelCOX, stmodelGLM, and steppes.

Old functions to be deprecated: stepp, stepp_summary, stepp_print, analyze.KM.stepp and analyze.CumInc.stepp.

Examples

```
# see example in the documentation for the function stepp.
```

`stepp_print`	*The function to print the estimate, covariance matrices and test statistics.*

Description

This function will be deprecated in the future. Please use S4 classes and generic functions summary, print and plot for future development.

A method to print three essential information resulting from the STEPP analysis.

Usage

```
stepp_print(x, estimate=TRUE, cov=TRUE, test=TRUE)
```

Arguments

`x`	a steppes object returned from stepp, analyze.KM.stepp or analyze.CumInc.stepp function
`estimate`	whether to print the effect estimates; default to yes
`cov`	whether to print the covariance matrices; default to yes
`test`	whether to print the test statistics; default to yes

Details

The STEPP analysis produces three important pieces of information. User can decide to print them all or individually. The three pieces of information are:

1. effect estimates of each subpopulation for the two treatments, the differences in absolute scale of the effects and the ratio in relative scale of the effects.
2. covariance matrics of the differences, logratios, differences in homogeneous association and logratios in homogeneous association.
3. various permutation p values based on test statistics: supremum pvalues, homogeneous association pvalues, chisquare pvalue. Not all statistics are available for all models.

Warning

This function together with other old functions will be depreciated in the future. A new set of S4 classes are implemented to replace old interfaces. Please use them for future development.

Author(s)

Wai-ki Yip, David Zahrieh, Marco Bonetti, Bernard Cole, Ann Lazar, Richard Gelber

See Also

The S4 classes: stwin, stsubpop, stmodelKM, stmodelCI, stmodelCOX, stmodelGLM, and steppes.

Old functions to be deprecated: stepp, stepp_summary, stepp_plot, analyze.CumInc.stepp and analyze.KM.stepp.

Examples

```
# see example in the documentation for the function stepp.
```

`stepp_summary` *The function to produce a summary of the size and various attributes of each subpopulation*

Description

This function will be deprecated in the future. Please use S4 classes and corresponding generic functions e.g. summary, print and plot for future development.

A method to print the summary of the size and various attributes for each subpopulation used in the STEPP analysis.

Usage

```
stepp_summary(x)
```

Arguments

`x`	a steppes object returned from stepp, analyze.KM.stepp or analyze.CumInc.stepp function

Warning

This function together with other old functions will be depreciated in the future. A new set of S4 classes are implemented to replace old interfaces. Please use them for future development.

Author(s)

Wai-ki Yip, David Zahrieh, Marco Bonetti, Bernard Cole, Ann Lazar, Richard Gelber

See Also

The S4 classes: stwin, stsubpop, stmodelKM, stmodelCI, stmodelCOX, stmodelGLM, and steppes.

Old functions to be deprecated: stepp, stepp_print, stepp_plot, analyze.KM.stepp, and analyze.CumInc.stepp

Examples

```
# set example in the documentation for the function stepp.
```

stmodel-class *Class* "stmodel"

Description

This is the S4 virtual class of all stepp models.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "stmodel" in the signature.

Author(s)

Wai-Ki Yip

See Also

stwin, stsubpop, stmodelKM, stmodelCI, stmodelGLM, steppes, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.GLM, stepp.test, estimate, generate

Examples

```
showClass("stmodel")
```

stmodelCI-class *Class* "stmodelCI"

Description

This is the stepp model of survival data with competing risks.

Value

The new method returns the stmodelCI object.

The estimate method returns a list with the following fields:

model	the stepp model - "CIe"
sObs1	a vector of effect estimates of all subpopulations based on the 1st treatment
sSE1	a vector of standard errors of effect estimates of all subpopulations based on the 1st treatment
oObs1	effect estimate of the entire population based on the 1st treatment

`oSE1`	the standard error of the effect estimate of the entire population based on the 1st treatment
`sObs2`	a vector of effect estimates of all subpopulations based on the 1st treatment
`sSE2`	a vector of standard errors of effect estimates of all subpopulations based on the 1st treatment
`oObs2`	effect estimate of the entire population based on the 1st treatment
`oSE2`	the standard error of the effect estimate of the entire population based on the 1st treatment
`skmw`	Wald's statistics for the effect estimate differences between the two treatments
`logHR`	a vector of log hazard ratio estimate of the subpopulations comparing 1st and 2nd treatments
`logHRSE`	a vector of standard error of the log hazard ratio estimate of the subpopulations comparing 1st and 2nd treatments
`ologHR`	the log hazard ratio estimate of the entire population comparing 1st and 2nd treatments
`ologHRSE`	the standard error of the log hazard ratio estimate of the entire population comparing 1st and 2nd treatments
`logHRw`	Wald's statistics for the log hazard ratio between the two treatments

The test method returns a list with the following fields:

`model`	the stepp model - "CIt"
`sigma`	the covariance matrix for subpopulations based on effect differences
`hasigma`	the homogeneous association covariance matrix for subpopulations based on effect differences
`HRsigma`	the covariance matrix for the subpopulations based on hazard ratio
`haHRsigma`	the homogeneous association covariance matrix for subpopulations based on hazard ratio
`pvalue`	the supremum pvalue based on effect difference
`chi2pvalue`	the chisquare pvalue based on effect difference
`hapvalue`	the homogeneous association pvalue based on effect difference
`HRpvalue`	the supremum pvalue based on hazard ratio
`haHRpvalue`	the homogeneous association pvalue based on hazard ratio

Objects from the Class

Objects can be created by calls of the form `new("stmodelCI", ...)` or by
the constructor function stepp.CI.

Slots

`coltrt`: Object of class `"numeric"`
: the treatment variable

`coltime`: Object of class `"numeric"`
: the time to event variable

`coltype`: Object of class `"numeric"`
: variable with distinct codes for different causes of failure where coltype=0 for censored observations; coltype=1 for event of interest; coltype=2 for other causes of failure

`trts`: Object of class `"numeric"`
: a vector containing the codes for the 2 treatment groups, 1st and 2nd treatment groups, respectively

`timePoint`: Object of class `"numeric"`
: timepoint to estimate survival

Extends

Class `"stmodel"`, directly.

Methods

estimate `signature(.Object = "stmodelCI")`:
: estimate the effect in absolute and relative scale of the overall and each subpopulation

print `signature(.Object = "stmodelCI")`:
: print the estimate, covariance matrices and statistics

test `signature(.Object = "stmodelCI")`:
: perform the permutation tests or GEE and obtain various statistics

Author(s)

Wai-Ki Yip

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
showClass("stmodelCI")

##
n <- 1000 # set the sample size
mu <- 0 # set the mean and sd of the covariate
sigma <- 1

beta0 <- log(-log(0.5)) # set the intercept for the log hazard
beta1 <- -0.2 # set the slope on the covariate
beta2 <- 0.5 # set the slope on the treatment indicator
```

```
beta3 <- 0.7 # set the slope on the interaction

prob2 <- 0.2 # set the proportion type 2 events
cprob <- 0.3 # set the proportion censored

set.seed(7775432)  # set the random number seed
covariate <- rnorm(n,mean=mu,sd=sigma) # generate the covariate values
Txassign <- rbinom(n,1,0.5) # generate the treatment indicator
x3 <- covariate*Txassign # compute interaction term
# compute the hazard for type 1 event
lambda1 <- exp(beta0+beta1*covariate+beta2*Txassign+beta3*x3)
lambda2 <- prob2*lambda1/(1-prob2) # compute the hazard for the type 2 event
# compute the hazard for censoring time
lambda0 <- cprob*(lambda1+lambda2)/(1-cprob)
t1 <- rexp(n,rate=lambda1) # generate the survival time for type 1 event
t2 <- rexp(n,rate=lambda2) # generate the survival time for type 2 event
t0 <- rexp(n,rate=lambda0) # generate the censoring time
time <- pmin(t0,t1,t2) # compute the observed survival time
type <- rep(0,n)
type[(t1 < t0)&(t1 < t2)] <- 1
type[(t2 < t0)&(t2 < t1)] <- 2

# create the stepp model object to analyze the data using Cumulative Incidence approach
x <- new ("stmodelCI", coltrt=Txassign, trts=c(0,1), coltime=time, coltype=type, timePoint=1.0)
```

stmodelGLM-class   *Class* "stmodelGLM"

Description

This is the stepp model for data arising from the following Generalized Linear Models: 1. gaussian with identity link, 2. binomial with logit link, and 3. Poisson with log link.

One can specify additional covariates in the model using `model.matrix` in R.

Value

The new method returns the stmodelGLM object.

The estimate method returns a list with the following fields:

`model`	the stepp model - "GLMGe" - Gaussian model, "GLMBe" - binomial model, and "GLMPe" - Poisson model
`sObs1`	a vector of effect estimates of all subpopulations based on the first treatment
`sSE1`	a vector of standard errors of effect estimates of all subpopulations based on the first treatment
`oObs1`	effect estimate of the entire population based on the first treatment
`oSE1`	the standard error of the effect estimate of the entire population based on the first treatment

`sObs2`	a vector of effect estimates of all subpopulations based on the first treatment
`sSE2`	a vector of standard errors of effect estimates of all subpopulations based on the first treatment
`oObs2`	effect estimate of the entire population based on the first treatment
`oSE2`	the standard error of the effect estimate of the entire population based on the first treatment
`sglmw`	Wald's statistics for the effect estimate differences between the two treatments
`RD`	a vector of effect estimates difference of all subpopulations between the two treatments
`RDSE`	a vector of the standard error effect estimates difference of all subpopulations between the two treatments
`ORD`	overall difference of effect estimates of the entire population between the two treatments
`ORDSE`	the standard error of the overall difference of the effect estimates of the entire population between the two treatments
`logR`	a vector of log ratio of effect estimates of all subpopulations between the two treatments
`logRSE`	a vector of standard error of log ratio of effect estimates of all subpopulations between the two treatments
`ologR`	log ratio of effect estimates of the entire population between the two treatments
`ologRSE`	the standard error of the log ratio of effect estimates of the entire population between the two treatments
`sglmlogrw`	Wald's statistics for the log ratio of effect estimates between the two treatments

The test method returns a list with the following fields:

`model`	the stepp model - "GLMt"
`sigma`	the covariance matrix for subpopulations based on effect differences
`hasigma`	the homogeneous association covariance matrix for subpopulations based on effect differences
`HRsigma`	the covariance matrix for the subpopulations based on hazard ratio
`haHRsigma`	the homogeneous association covariance matrix for subpopulations based on hazard ratio
`pvalue`	the supremum pvalue based on effect difference
`chi2pvalue`	the chisquare pvalue based on effect difference
`hapvalue`	the homogeneous association pvalue based on effect difference
`HRpvalue`	the supremum pvalue based on hazard ratio
`haHRpvalue`	the homogeneous association pvalue based on hazard ratio

Objects from the Class

Objects can be created by calls of the form `new("stmodelGLM", ...)` or by
the construction function stmodel.GLM.

Slots

`coltrt`**:** Object of class `"numeric"`
: the treatment variable

`colY`**:** Object of class `"numeric"`
: a vector containing the outcome

`trts`**:** Object of class `"numeric"`
: a vector containing the codes for the 2 treatment groups, first and second treatment groups, respectively

`MM`**:** Object of class `"ANY"`
: a model matrix for extra adjustment covariates; default is NULL
currently, stepp can only support covariates with numeric values, no logical values or factors allowed
these values need to be converted to numeric with some encoding schemes

`glm`**:** Object of class `"character"`
: the glm to be used for analysis: "gaussian", "binomial", "poisson"

`link`**:** Object of class `"character"`
: the link function; reserved for future use

Extends

Class `"stmodel"`, directly.

Methods

estimate `signature(.Object = "stmodelGLM")`:
: estimate the effect in absolute and relative scale of the overall and each subpopulation

print `signature(.Object = "stmodelGLM")`:
: print the estimate, covariance matrices and statistics

test `signature(.Object = "stmodelGLM")`:
: perform the permutation tests or GEE and obtain various statistics

Author(s)

Wai-Ki Yip

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
showClass("stmodelGLM")
```

`stmodelKM-class` *Class* `"stmodelKM"`

Description

This is the S4 class for the stepp model of survival data using Kaplan-Meier method.

Value

The new method returns the stmodelKM object.

The estimate method returns a list with the following fields:

`model`	the stepp model - "KMe"
`sObs1`	a vector of effect estimates of all subpopulations based on the first treatment
`sSE1`	a vector of standard errors of effect estimates of all subpopulations based on the first treatment
`oObs1`	effect estimate of the entire population based on the first treatment
`oSE1`	the standard error of the effect estimate of the entire population based on the first treatment
`sObs2`	a vector of effect estimates of all subpopulations based on the group treatment
`sSE2`	a vector of standard errors of effect estimates of all subpopulations based on the first treatment
`oObs2`	effect estimate of the entire population based on the first treatment
`oSE2`	the standard error of the effect estimate of the entire population based on the first treatment
`skmw`	Wald's statistics for the effect estimate differences between the two treatments
`logHR`	a vector of log hazard ratio estimate of the subpopulations comparing first and second treatments
`logHRSE`	a vector of standard error of the log hazard ratio estimate of the subpopulations comparing first and second treatment
`ologHR`	the log hazard ratio estimate of the entire population comparing first and second treatment
`ologHRSE`	the standard error of the log hazard ratio estimate of the entire population comparing first and second treatment
`logHRw`	Wald's statistics for the log hazard ratio between the two treatment

The test method returns a list with the following fields:

`model`	the stepp model - "KMt"
`sigma`	the covariance matrix for subpopulations based on effect differences
`hasigma`	the homogeneous association covariance matrix for subpopulations based on effect differences

`HRsigma`	the covariance matrix for the subpopulations based on hazard ratios
`haHRsigma`	the homogeneous association covariance matrix for subpopulations based on hazard ratios
`pvalue`	the supremum pvalue based on effect difference
`chi2pvalue`	the chisquare pvalue based on effect difference
`hapvalue`	the homogeneous association pvalue based on effect difference

Objects from the Class

Objects can be created by calls of the form `new("stmodelKM", ...)` or by
the constructor function stmodel.KM.

Slots

`coltrt`**:** Object of class `"numeric"`
the treatment variable

`survTime`**:** Object of class `"numeric"`
the time to event variable

`censor`**:** Object of class `"numeric"`
the censor variable

`trts`**:** Object of class `"numeric"`
a vector containing the codes for the 2 treatment groups, first and second treatment groups, respectively

`timePoint`**:** Object of class `"numeric"`
timepoint to estimate survival

Extends

Class `"stmodel"`, directly.

Methods

estimate `signature(.Object = "stmodelKM")`:
estimate the effect in absolute and relative scale of the overall population and each subpopulation.

print `signature(.Object = "stmodelKM")`:
print the estimate, covariance matrices and statistics.

test `signature(.Object = "stmodelKM")`:
perform the permutation tests or GEE and obtain various statistics.

Author(s)

Wai-Ki YIp

See Also

`stwin`, `stsubpop`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
showClass("stmodelKM")

#GENERATE TREATMENT VARIABLE:
N <- 1000
Txassign <- sample(c(1,2), N, replace=TRUE, prob=c(1/2, 1/2))
n1 <- length(Txassign[Txassign==1])
n2 <- N - n1
#GENERATE A COVARIATE:
covariate <- rnorm(N, 55, 7)
#GENERATE SURVIVAL AND CENSORING VARIABLES ASSUMING A TREATMENT COVARIATE INTERACTION:
Entry <- sort( runif(N, 0, 5) )
SurvT1 <- .5
beta0 <-  -65 / 75
beta1 <- 2 / 75
Surv <- rep(0, N)
lambda1 <- -log(SurvT1) / 4
Surv[Txassign==1] <- rexp(n1, lambda1)
Surv[Txassign==2] <- rexp(n2, (lambda1*(beta0+beta1*covariate[Txassign==2])))
EventTimes <- rep(0, N)
EventTimes <- Entry + Surv
censor <- rep(0, N)
time <- rep(0,N)
for ( i in 1:N )
   {
     censor[i] <- ifelse( EventTimes[i] <= 7, 1, 0 )
     time[i] <- ifelse( EventTimes[i] < 7, Surv[i], 7 - Entry[i] )
   }

modKM <- new("stmodelKM", coltrt=Txassign, survTime=time, censor=censor, trts=c(1,2), timePoint=4)
```

---

stsubpop-class *Class* "stsubpop"

Description

This is the S4 class for stepp subpopulation object. The subpopulations are generated based on the stepp windows and the covariate of interest.

Objects from the Class

Objects can be created by calls of the form `new("stsubpop")` or the constructor method stepp.subpop.

Slots

`win`: Object of class "stwin"
the stepp window set up for the analysis

`covar`**:** Object of class `"numeric"`
: the covariate of interest

`nsubpop`**:** Object of class `"numeric"`
: the number of subpopulations generated

`subpop`**:** Object of class `"ANY"`
: a matrix of subpopulations generated based on the stepp window and the specified covariate of interest

`npatsub`**:** Object of class `"numeric"` a vector of size of each subpopulation

`medianz`**:** Object of class `"numeric"`
: a vector of median value of the covariate of interest for each subpopulation

`minc`**:** Object of class `"numeric"`
: a vector of the minimum value of the covariate of interest for each subpopulation

`maxc`**:** Object of class `"numeric"`
: a vector of the maximum value of the covariate of interest for each subpopulation

`neventsubTrt0`**:** Object of class `"numeric"` or `NULL`
: a vector containing the number of events in each subpopulation for the baseline treatment group

`neventsubTrt1`**:** Object of class `"numeric"` or `NULL`
: a vector containing the number of events in each subpopulation for the active treatment group

`init`**:** Object of class `"logical"`
: a logical value indicating if the subpopulations have already been generated or not

Methods

generate `signature(.Object = "stsubpop", win, covariate, coltype, coltrt, trts, minsubpops)`:
: a method to generate the subpopulations based on the stepp window object and the specified covariate of interest. For event-based windows, also the event type (`coltype`), treatment indicator (`coltrt`), treatments list (`trts`) and minimum number of subpopulations (`minsubpops`) must be provided

summary `signature(.Object = "stsubpop")`:
: a method to display the summary of the subpopulations generated

Author(s)

Wai-Ki Yip

See Also

`stwin`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
showClass("stsubpop")

# create a steppp window
win1 <- stepp.win(type="sliding", r1=5,r2=10)
```

```
# generate the covariate of interest
Y <- rnorm(100)

# create and generate the stepp subpopulation
sp <- new("stsubpop")
sp <- generate(sp, win=win1, cov=Y)
summary(sp)

# event-based windows using the BIG data set
data(bigKM)
rxgroup <- bigKM$trt
time    <- bigKM$time
evt     <- bigKM$event
cov     <- bigKM$ki67

swin_e <- new("stwin", type = "sliding_events", e1 = 10, e2 = 20)
subp_e <- new("stsubpop")
subp_e <- generate(subp_e, win = swin_e, covariate = cov, coltype = evt,
                   coltrt = rxgroup, trts = c(1, 2), minsubpops = 5)
summary(subp_e)
```

stwin-class   *Class* "stwin"

Description

This the S4 class for the stepp window object. The stepp window class describes the way to set up the subpopulation for a stepp analysis. It specifies a pattern of subpopulation you would like to explore. There are three kinds of patterns: "sliding window", "event-based sliding window" and "tail-oriented window".

These patterns are specified through the following set of values (r1 and r2 for sliding and tail-oriented windows, e1 and e2 for event-based sliding windows):

1. for sliding windows based on the number of patients ("sliding"), r1 is the minimum number of patients allowed in overlapping windows and r2 is the approximate size of subpopulation in each window;

2. for sliding windows based on the number of events ("sliding_events"), e1 is the minimum number of events allowed in overlapping windows and e2 is the approximate size of subpopulation in each window in terms of number of events;

3. for tail-oriented window, r1 is a vector of maximum covariate value for each subpopulation from the minimum value of the entire subpopulation, and r2 is a vector of minimum covariate values for each subpopulation from the maximum value of the window. The utility function `gen.tailwin()` can be used to generate these vectors based on the number of desired subpopulations.

When "sliding_events" is chosen r1 and r2 are set to `NULL`, while when "sliding" or "tail-oriented" are chosen e1 and e2 are set to `NULL`.

These pattern are based on all patients.

Objects from the Class

Objects can be created by calls of the form `new("stwin", type="sliding", r1=5, r2=20)` or the constructor function `stepp.win`.

Slots

`type`**:** Object of class `"character"`
: stepp window type; `"sliding"`, `"sliding_events"` or `"tail-oriented"`

`r1`**:** Object of class `"numeric"` or `NULL`
: sliding window: minimum number of patients allowed in overlapping windows;
: tail-oriented window: a vector of maximum covariate value for each subpopulation

`r2`**:** Object of class `"numeric"` or `NULL`
: sliding window: size of subpopulation in each window;
: tail-oriented window: a vector of minimum covariate value for each subpopulation

`e1`**:** Object of class `"numeric"` or `NULL`
: event-based sliding window: minimum number of events allowed in overlapping windows

`e2`**:** Object of class `"numeric"` or `NULL`
: event-based sliding window: number of events in each subpopulation

Methods

summary `signature(object = "stwin")`:
: print a summary of the stepp windows object

Author(s)

Wai-Ki Yip

See Also

`stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `stepp.test`, `estimate`, `generate`

Examples

```
showClass("stwin")

# create a stepp window of type "sliding" with (r2) size of subpopulation
# in each window to be 200 and (r1) allows only 50 patients in the
# overlapping windows
mywin <- new("stwin", type="sliding", r1=50, r2=200)

# print a summary of the stepp window object created
summary(mywin)
```

```
# create a stepp window object of type "sliding_events",
# (event-based) subpopulation size is 200 and allows
# only 50 events between overlapping windows
mywin <- stepp.win(type="sliding_events", e1=50, e2=200)

# print a summary of the stepp window object
summary(mywin)
```

---

test *the standard generic function for all test methods*

Description

The default generic function for test methods for all stepp models classes and the steppes class.

For detail, please refer to the documentation in the test method in the S4 class: stmodelCI, stmodelKM and stmodelGLM.

Author(s)

Wai-ki Yip

See Also

`stwin`, `stsubpop`, `stmodelKM`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.GLM`, `generate`, `estimate`

Index