Package ‘stepp’

February 20, 2015

Type Package
Title Subpopulation Treatment Effect Pattern Plot (STEPP)
Version 3.0-11
Date 2013-Dec-8
Author Wai-ki Yip, with contributions from
Ann Lazar, David Zahrieh, Chip Cole, Ann Lazar, Marco Bonetti,
Victoria Wang, William Barcella and Richard Gelber
Maintainer Wai-ki Yip <wkyip@hsph.harvard.edu>
Depends methods, car, survival, splines
Collate 'stutil.R' 'stwin.R' 'stsubpop.R' 'stmodelKM.R' 'stmodelCOX.R'
' stmodelCL.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 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 http://www.r-project.org
LazyLoad yes
NeedsCompilation yes
Repository CRAN
Date/Publication 2014-10-22 09:03:30

R topics documented:

analyze.CumInc.stepp .................................................. 2
analyze.KM.stepp ...................................................... 6
aspirin ........................................................................ 9
big .............................................................................. 11
estimate ....................................................................... 13
generate ..................................................................... 13
stepp .......................................................................... 14
analyze competing risks data using Cumulative Incidence method

Description

This method will be deprecated in the future. Please use S4 classes e.g. stmodelCI 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, colvar, trts, patspop, minpatspop, timest, noperm=2500, ncex = 0.7, legeny = 30, pline = -2.5, color = c("red", "black"),
xlabel = "Subpopulations by Median Covariate",
ylabel = "?‐year Disease‐Free Survival",
legend = 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
- **colvar** 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 nperm to 0. In that case, pvalue and the covariance matrix will not be available.

We acknowledge Robert J. Gray for permitting us to use the cmprsk package.
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 colvar 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.
analyze.CumInc.stepp

**Author(s)**

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

**References**


**See Also**

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

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

**Examples**

```r
#GENERATE TREATMENT VARIABLE:
set.seed(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 the random number seed
set.seed(1000)

# generate the covariate values
covariate <- rnorm(n, mean = mu, sd = sigma)

# generate the treatment indicator
Txassign <- rbinom(n, 1, 0.5)

# compute interaction term
x3 <- covariate * Txassign

# compute the hazard for type 1 event
lambda1 <- exp(beta0 + beta1 * covariate + beta2 * Txassign + beta3 * x3)

# compute the hazard for type 2 event
lambda2 <- prob2 * lambda1 / (1 - prob2)

# 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

# set the event type

# generate event type

# generate event type
```
analyze.KM.stepp

analyze survival data using Kaplan-Maier method

Description

This method will be deprecated in the future. Please use S4 classes e.g. stmodelKM 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, colvar, trts, patspop, minpatspop, timest, noperm=2500, ncex = 0.7, legendy = 30, pline = -2.5, color = c("red", "black"), xlabel = "Subpopulations by Median Covariate", ylabel = "7-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
colvar the covariate of interest
trts a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment arms, respectively
analyze.KM.stepp

- `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
- `legeny`: 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 covariance 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 colvar 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


See Also

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

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, colvar=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 is the data from the Aspirin/Folate Polyp Prevention Study Group collaborative investigating the effect of aspirin on the outcome of any occurrence of a colorectal adenoma and any occurrence of an advanced lesion. There are 1,121 subjects randomized to three doses of aspirin (placebo, 81 mg/day and 325 mg/day). The subjects were followed for three years and then underwent colonoscopy. Follow-up information is available for 1,084 subjects. The remainder did not undergo a follow-up exam.

The data set contained here is a "simulated" data set based on the actual data from the Aspirin/Folate Polyp Prevention Study.

Usage

data(aspirin)

Format

The data set has the following columns: ID - Patient ID number DOSE - Randomized aspirin dose (0=placebo) AGE - Age of the patient HT - Height (in meters) WT - Weight (in Kg) G - Gender (M or F) AD - Any adenoma (0=no, 1=yes, .=no exam) LE - Any advanced lesion (0=no, 1=yes, .=no exam)

Source

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

References


Examples

data(aspirin)

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

# model matrix can only handle numeric data
# convert to 0, 1 encoding for gender
GENDER <- as.numeric( aspirin[,"G"] == "M")
aspirin <- cbind(aspirin, GENDER)
remove("GENDER")

# make a subset of patients with placebo and 81 mg
attach(aspirin)
subset1 <- DOSE == 0 | DOSE == 81
aspirin1 <- aspirin[subset1,]
detach(aspirin)
# set up treatment assignment
trtA <- rep(0, dim(aspirin)[1])
trtA[aspirin[1,"DOSE"] == 81] <- 1

# STEPP analysis A: placebo vs 81 mg aspirin
attach(aspirin)
inc_win <- stepp.win(type="sliding", r1=30, r2=100)
inc_sp <- stepp.subpop(swin=inc_win, cov=AGE)
ADorLE <- as.numeric(AD | LE)
modelA <- stepp.GLM(coltrt=trtA, trts=c(0,1), colY=ADorLE, glm="binomial", debug=0)
# 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(aspirin)

big  the big data set.

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 ther-
apy 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.

The data set contained here is a "simulated" data set based on the actual data from the BIG 1-98
clinical trial.

Usage

data(big)

Format

There are four columns of numeric values: rxgroup (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


Examples

data(big)

rxgroup <- big$rxgroup
time <- big$time
evt <- big$event
cov <- big$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, nce=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("Tamoxifen", "Letrozole"), 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.

Author(s)

Wai-ki Yip

See Also

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

generate

the standard generic function for the generate method in stsubpop class

Description

The default generic function for generate method for stsubpop class.

Author(s)

Wai-ki Yip

See Also

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

This method will be deprecated in the future. Please use S4 classes e.g. stmodelCI 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, colvar, 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

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

colvar: 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 covariance 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 analysis 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 deprecated 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 colrt, coltime, coltype, and colvar 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


See Also

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

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

Examples

```r
# GENERATE TREATMENT VARIABLE:
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
```
\[
\lambda_1 = \exp(\beta_0 + \beta_1 + \text{covariate} + \beta_2 + \text{Txassign} + \beta_3 \times x_3)
\]
\[
\lambda_2 = \frac{\text{prob_2} \times \lambda_1}{(1 - \text{prob_2})} \quad \# \text{compute the hazard for the type 2 event}
\]
\[
\lambda_0 = \frac{\text{cprob} \times (\lambda_1 + \lambda_2)}{(1 - \text{prob})} \quad \# \text{compute the hazard for censoring time}
\]
\[
t_1 = \text{rexp}(n, \text{rate} = \lambda_1) \quad \# \text{generate the survival time for type 1 event}
\]
\[
t_2 = \text{rexp}(n, \text{rate} = \lambda_2) \quad \# \text{generate the survival time for type 2 event}
\]
\[
t_0 = \text{rexp}(n, \text{rate} = \lambda_0) \quad \# \text{generate the censoring time}
\]
\[
time = \text{pmin}(t_0, t_1, t_2) \quad \# \text{compute the observed survival time}
\]
\[
type = \text{rep}(0, \text{n})
\]
\[
type[[t_1 < t_0 \& (t_1 < t_2)]] <- 1
\]
\[
type[[t_2 < t_0 \& (t_2 < t_1)]] <- 2
\]

# 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, colvar = covariate, patspop = 300, minpatspop = 200, timest = 1.0, noperm = 0)

stepp_summary(steppCI)
stepp_print(steppCI)
stepp_plot(steppCI,
  nce = 0.70, legend = 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

---

**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.

**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 arms, 1st and 2nd treatment arms, respectively
- timePoint: timepoint to estimate survival
Value

It returns the stmodelCI object.

Author(s)

Wai-Ki Yip

See Also

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

Examples

```r
##
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

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

stepp.COX  a method to create the stmodelCOX object

Description

This is the constructor function for the stmodelCOX object. This object sets up the data with a stepp model using Cox method for analysis.

Usage

stepp.COX(coltrt, survTime, censor, trts, timePoint, MM)

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 arms, respectively
timePoint timepoint to estimate survival
MM a model matrix for any additional covariates; default is NULL

Details

One can specify additional covariates for adjustment when using the COX PH model using model.matrix in R. mm <- model.matrix(~cov1+cov2+...+covn). Since there is no intercept term in the COX model, the intercept column will be automatically stripped if it is used.

Value

It returns a stmodelCOX object.

Author(s)

Wai-Ki Yip

See Also

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

```r
# 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] )
}
c1 <- rnorm(1000)

# create the model matrix to represent extra covariate for adjustment
mm0 <- model.matrix(~c1)

# create the stepp model for survival data using COX PH model for analysis
modCOX <- stepp.COX(coltrt=Txassign, survTime=time, censor=censor, trts=c(1,2), timePoint=4, MM=mm0)
```
Arguments

`coltr`  the treatment variable

`trts`  a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment arms, 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"

`debug`  for internal debugging use only

Value

It returns a stmodelGLM object.

Author(s)

Wai-Ki Yip

See Also

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

Examples

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

---

**stepp.KM**  
*a method 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 Kaplan-Meier method for analysis.

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 arms, respectively
- `timePoint` - timepoint to estimate survival

Value

It returns the stmodelKM object.

Author(s)

Wai-Ki Yip

See Also

- `stwin`, `stsubpop`, `stmodelKM`, `stmodelCOX`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.CI`, `stepp.COX`, `stepp.GLMM`, `stepp.test`, `estimate`, `generate`

Examples

```r
#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.subpop

---

**stepp.subpop**

*a method 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)`

**Arguments**

- `swin` the stepp window set up for the analysis
- `cov` the covariate of interest

**Value**

It returns the stsubpop object with subpopulations generated.

**Author(s)**

Wai-Ki Yip

**See Also**

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

**Examples**

```r
# create a stepp 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)
```

stepp.test  

*a method 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

```r
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 by Bonetti below). For best results, considering using 2500 permutations to obtain a rich distribution in which to draw inference.

Value

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

Author(s)

Wai-ki Yip

References


See Also

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

Examples

data(big)

rxgroup <- big$rxgroup
time <- big$time
evt <- big$event
cov <- big$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

a method to create the stepp window object

Description

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

Usage

stepp.win(type, r1, r2)

Arguments

type type of stepp window; only "sliding" is supported
r1 minimum number of patients allowed in overlapping windows
r2 size of subpopulation in each window

Details

This is the functional interface to construct a stepp window object besides using new("stwin", ...).

Author(s)

Wai-ki Yip
See Also

stwin, stsubpop, stmodelKM, stmodelCOX, stmodelCI, stmodelGLM, steppes, stmodel, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, 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)

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
result: Object of class "ANY"
    permutation and 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"): estimate the effect in absolute and relative scale of the overall and each subpopulation

**plot** signature(x = "steppes"): generate the three stepp plots

**print** signature(x = "steppes"): 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"): perform the permutation tests and obtain various statistics

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, stmodelCOX, stmodelCI, stmodelGLM, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, stepp.GLM, stepp.test, estimate, generate

Examples

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

---

stepp_plot generating the stepp plots

Description

This method 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, diff = TRUE, ci = TRUE, pv = TRUE,
showss = TRUE, ylim = c(0, 100, -100, 100, 0, 3), dev = "", together = FALSE,
noyscale = FALSE, at = NA)

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 parameter (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

diff optional - specify if you want just the first plot; default is TRUE (all three plots)

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

ylim optional - pvalue will be displayed in the plots; FALSE: pvalue will not be displayed in the plots; default is TRUE

dev optional - output device control. Output the plots to the specified output format. postscript for .ps; eps (encapsulated postscript) for .eps; pdf for .pdf; png for .png; bmp for .bmp; tiff for .tiff and jpeg for .jpeg. The filenames are always SteppPlot1, SteppPlot2 and SteppPlot3 in the R working directory; default is ""
**stepp_plot**

- **together**: optional - output the plots to a single file instead of 3 separate files; default is FALSE
- **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

**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. The output is usually directed to the R console output and will appear in separate windows. You can also direct the plots to various files in different formats: postscript for .ps; eps (encapsulated postscript) for .eps; pdf for .pdf; png for .png; bmp for .bmp; tiff for .tiff and jpeg for .jpeg through the use of the dev parameter. The filenames are always SteppPlot1, SteppPlot2 and SteppPlot3 in the R working directory. You can also direct the plots to one file only by setting the together parameter to TRUE.

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  

*a method to print the estimate, covariance matrices and test statistics.*

**Description**

This method 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 matrices 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 deprecated 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.
stepp_summary

Examples

# see example in the documentation for the function stepp.

stepp_summary produce a summary of the size and various attributes of each subpopulation

Description

This method 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, stmodelCOX, stmodelCI, stmodelGLM, steppes, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, stepp.GLMM, 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"
- s0bs1: 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
- o0bs1: effect estimate of the entire population based on the 1st treatment
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 arms, 1st and 2nd treatment arms, 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(x = "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, stmodelCOX, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win,
stepp.subpop, stepp.KM, stepp.COX, 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
tassign <- rbinom(n,1,0.5) # generate the treatment indicator
x3 <- covariate*tassign # compute interaction term
# compute the hazard for type 1 event
lambda1 <- exp(beta0+beta1*covariate+beta2*tassign+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=tassign, trts=c(0,1), coltime=time, coltype=type, timePoint=1.0)

---

**stmodelCOX-class**  
*Class "stmodelCOX"*

**Description**

This is the stepp model for survival data using COX proportional hazard

**Value**

The new method returns the stmodelCOX object.

The estimate method returns a list with the following fields:

- **model**: the stepp model - "COXe"
- **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
effect estimate of the entire population based on the 1st treatment
the standard error of the effect estimate of the entire population based on the 1st treatment
Wald’s statistics for the effect estimate differences between the two treatments
a vector of log hazard ratio estimate of the subpopulations comparing 1st and 2nd treatments
a vector of standard error of the log hazard ratio estimate of the subpopulations comparing 1st and 2nd treatments
the log hazard ratio estimate of the entire population comparing 1st and 2nd treatments
the standard error of the log hazard ratio estimate of the entire population comparing 1st and 2nd treatments
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 - "COXt"
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
GEEpvalue the GEE based pvalue

Objects from the Class

Objects can be created by calls of the form new("stmodelCOX", ...) or by the construction function stmodel.COX.

Slots

mm: Object of class "ANY"
a model matrix created using model.matrix in R to specify additional covariates for adjustment; default is NULL
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 arms, 1st and 2nd treatment arms, respectively

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

Extends
Class "stmodelKM", directly. Class "stmodel", directly.

Methods

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

print signature(x = "stmodelCOX"):
print the estimate, covariance matrices and statistics

test signature(.Object = "stmodelCOX"):
perform the permutation tests and obtain various statistics

Author(s)
Wai-Ki Yip

See Also
stwin, stsubpop, stmodelKM, stmodelCOX, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.GL, stepp.test, estimate, generate

Examples

showClass("stmodelCOX")
#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])
}

c1 <- rnorm(1000)

# create the model matrix to represent extra covariate for adjustment
mm0 <- model.matrix(~c1)

# create the stepp model for survival data using COX PH model for analysis
modCOX <- new("stmodelCOX", coltrt=Tassign, survtime=time, censor=censor, trts=c(1,2),
              timePoint=4, MM=mm0)

---

**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 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
Wald’s statistics for the effect estimate differences between the two treatments

a vector of effect estimates difference of all subpopulations between the two treatments

a vector of the standard error effect estimates difference of all subpopulations between the two treatments

overall difference of effect estimates of the entire population between the two treatments

the standard error of the overall difference of the effect estimates of the entire population between the two treatments

a vector of log ratio of effect estimates of all subpopulations between the two treatments

a vector of standard error of log ratio of effect estimates of all subpopulations between the two treatments

log ratio of effect estimates of the entire population between the two treatments

the standard error of the log ratio of effect estimates of the entire population between the two treatments

Wald’s statistics for the log ratio of effect estimates between the two treatments

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 arms, 1st and 2nd treatment arms, 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

ddebug: Object of class "numeric"

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(x = "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, stmodelCOX, stmodelCI, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, stepp.GLM, stepp.test, estimate, generate

Examples
showClass("stmodelGLM")
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"
- **s0bs1**: 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
- **o0bs1**: 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
- **s0bs2**: 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
- **o0bs2**: 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 - "KMt"
- **sigma**: the covariance matrix for subpopulations based on effect differences
- **hasigma**: the homogeneous association covariance matrix for subpopulations based on effect differences
\textit{stmodelKM-class}

\begin{itemize}
\item \texttt{HRsigma} \hspace{1cm} \text{the covariance matrix for the subpopulations based on hazard ratio}
\item \texttt{haHRsigma} \hspace{1cm} \text{the homogeneous association covariance matrix for subpopulations based on hazard ratio}
\item \texttt{pvalue} \hspace{1cm} \text{the supremum pvalue based on effect difference}
\item \texttt{chi2pvalue} \hspace{1cm} \text{the chisquare pvalue based on effect difference}
\item \texttt{hapvalue} \hspace{1cm} \text{the homogeneous association pvalue based on effect difference}
\item \texttt{HRpvalue} \hspace{1cm} \text{the supremum pvalue based on hazard ratio}
\item \texttt{hahrpvalue} \hspace{1cm} \text{the homogeneous association pvalue based on hazard ratio}
\end{itemize}

\textbf{Objects from the Class}

Objects can be created by calls of the form \texttt{new("stmodelKM", \ldots)} or by the constructor function \texttt{stmodel.KM}.

\textbf{Slots}

\begin{itemize}
\item \texttt{coltrt} \hspace{1cm} \text{Object of class "numeric"}
\hspace{1cm} \text{the treatment variable}
\item \texttt{survTime} \hspace{1cm} \text{Object of class "numeric"}
\hspace{1cm} \text{the time to event variable}
\item \texttt{censor} \hspace{1cm} \text{Object of class "numeric"}
\hspace{1cm} \text{the censor variable}
\item \texttt{trts} \hspace{1cm} \text{Object of class "numeric"}
\hspace{1cm} \text{a vector containing the codes for the 2 treatment arms, 1st and 2nd treatment arms, respectively}
\item \texttt{timePoint} \hspace{1cm} \text{Object of class "numeric"}
\hspace{1cm} \text{timepoint to estimate survival}
\end{itemize}

\textbf{Extends}

Class \"stmodel\", directly.

\textbf{Methods}

\begin{itemize}
\item \texttt{estimate} \hspace{1cm} \text{signature(.Object = "stmodelKM")}
\hspace{1cm} \text{estimate the effect in absolute and relative scale of the overall population and each subpopulation.}
\item \texttt{print} \hspace{1cm} \text{signature(x = "stmodelKM")}
\hspace{1cm} \text{print the estimate, covariance matrices and statistics.}
\item \texttt{test} \hspace{1cm} \text{signature(.Object = "stmodelKM")}
\hspace{1cm} \text{perform the permutation tests or GEE and obtain various statistics.}
\end{itemize}

\textbf{Author(s)}

Wai-Ki YIp
stsubpop-class

See Also

stwin, stsubpop, stmodelCOX, stmodelCI, stmodelGLM, steppes, stepp.model, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, 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 window 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
- colvar: 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 the median value of the covariate of interest for each subpopulation
- minz: Object of class "numeric"
  a vector of the minimum value of the covariate of interest for each subpopulation
- maxz: Object of class "numeric"
  a vector of the maximum value of the covariate of interest for each subpopulation
- init: Object of class "logical"
  a logical value indicating if the subpopulations are generated or not

Methods

- generate signature(.Object = "stsubpop"):
  a method to generate the subpopulations based on the stepp window and specified covariate of interest
- summary signature(object = "stsubpop"):
  a method to display the summary of the subpopulations generated

Author(s)

Wai-Ki Yip

See Also

stwin, stmodelKM, stmodelCOX, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, stepp.GL, 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)
stwin-class  

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

---

### 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. Each window represents a subpopulation along the covariate of interest. The only supported window type is "sliding" which means subpopulations overlap one another as the analysis is done along the covariate of interest. Two parameters control how the subpopulations are constructed: r2 specifies the size of each subpopulation and r1 specifies the minimum number of patients allowed to overlap between two adjacent subpopulations.

### 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; currently only support "sliding"
- **r1**: Object of class "numeric"
  - minimum number of patients allowed to overlap between subpopulations
- **r2**: Object of class "numeric"
  - size of each subpopulation
- **nevent**: Object of class "numeric"
  - reserve for future use

### Methods

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

### Author(s)

Wai-Ki Yip

### See Also

- `stsubpop`, `stmodelKM`, `stmodelCOX`, `stmodelCI`, `stmodelGLM`, `steppes`, `stmodel`, `stepp.win`, `stepp.subpop`, `stepp.KM`, `stepp.CI`, `stepp.COX`, `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)

description

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

Author(s)

Wai-ki Yip

See also

stwin, stsubpop, stmodelKM, stmodelCOX, stmodelCI, stmodelGLM, steppes, stmodel, stepp.win, stepp.subpop, stepp.KM, stepp.CI, stepp.COX, stepp.GL, generate, estimate
Index

*Topic \textasciitilde kwd1
  analyze.CumInc.stepp, 2
  analyze.KM.stepp, 6
  estimate, 13
  generate, 13
  stepp, 14
  stepp.CI, 17
  stepp.COX, 19
  stepp.GLM, 20
  stepp.KM, 21
  stepp.subpop, 23
  stepp.test, 24
  stepp.win, 25
  stepp_plot, 27
  stepp_print, 30
  stepp_summary, 31
  test, 46

*Topic \textasciitilde kwd2
  analyze.CumInc.stepp, 2
  analyze.KM.stepp, 6
  generate, 13
  stepp, 14
  stepp.CI, 17
  stepp.COX, 19
  stepp.GLM, 20
  stepp.KM, 21
  stepp.subpop, 23
  stepp.test, 24
  stepp.win, 25
  stepp_plot, 27
  stepp_print, 30
  stepp_summary, 31
  test, 46

*Topic \textasciitilde classes
  steppes-class, 26
  stmodel-class, 32
  stmodelCI-class, 32
  stmodelCOX-class, 35
  stmodelGLM-class, 38
  stmodelKM-class, 41
  stsubpop-class, 43
  stwin-class, 45

*Topic \textasciitilde datasets
  aspirin, 9
  big, 11

  analyze.CumInc.stepp, 2
  analyze.KM.stepp, 6
  aspirin, 9
  big, 11
  estimate, 13, 18, 19, 21–24, 26, 27, 32, 34, 37, 40, 43–46
  estimate, steppes-method (steppes-class), 26
  estimate, stmodelCI-method (stmodelCI-class), 32
  estimate, stmodelCOX-method (stmodelCOX-class), 35
  estimate, stmodelGLM-method (stmodelGLM-class), 38
  estimate, stmodelKM-method (stmodelKM-class), 41
  generate, 13, 18, 19, 21–24, 26, 27, 32, 34, 37, 40, 43–46
  generate, stsubpop-method (stsubpop-class), 43
  plot, steppes-method (steppes-class), 26
  print, steppes-method (steppes-class), 26
  print, stmodelCI-method (stmodelCI-class), 32
  print, stmodelCOX-method (stmodelCOX-class), 35
  print, stmodelGLM-method (stmodelGLM-class), 38