Package 'paf'

February 20, 2015

Title Attributable Fraction Function for Censored Survival Data

Version 1.0

Depends R(>= 2.13.0), survival

Date 2014-01-23

Author Li Chen

Maintainer Li Chen <lichenuky@uky.edu>

NeedsCompilation no

Repository CRAN

Description Calculate unadjusted/adjusted attributable fraction function of a set of covariates for a censored survival outcome from a Cox model using the method proposed by Chen, Lin and Zeng (Biometrika 97, 713-726., 2010).

License GPL-2

Date/Publication 2014-02-05 16:55:54

R topics documented:

`paf` *Calculate attributable fraction function for censored survival data*

Description

Fit a Cox model and calculate the unadjusted/adjusted attributable fraction function of a set of covariates based on the Cox model using the method proposed by Chen, Lin and Zeng (2010).

Usage

```
paf(formula, data, cov)
```

Arguments

formula	a formula object for the Cox model considered , which has the same format as that in the coxph function of the survival package.
data	a data.frame in which to interpret the variables named in the formula.
cov	the set of covariates whose attributable fraction function is of interest.

Details

This function calculates the unadjusted/adjusted attributable fraction function for the set of covariates specified in cov which must also be included as covariates of the Cox model. The function calculates the unadjusted attributable fraction function if the Cox model does not include other covariates; otherwise the function calculates the adjusted attributable fraction function adjusting for other covariates in the Cox model.

Value

time	unique uncensored event times at which the attributable fraction function jumps.
est	the estimates of unadjusted/adjusted attributable fractions at unique uncensored event times.
se	the standard errors of the estimated attributable fractions.
low	the lower confidence limits of the atrtributable fractions.
upp	the upper confidence limits of the atrtributable fractions.
fit.cox	coxph object from the fitted Cox model.

Note

The Breslow method is used to handle ties. The function will do missing-data filter automatically.

Author(s)

Li Chen

References

Chen L, Lin DY, Zeng D. (2010). Attributable fraction functions for censored event times. *Biometrika* **97**, 713-726.

See Also

plot.paf.

Examples

```
# simulated data set from a Cox model
n = 1000
x1 = as.numeric(runif(n)>0.5)
x2 = x1 + rnorm(n)
t = exp(-x1 - 0.5 * x2) * rexp(n, rate = 0.1)
c = runif(n, 0, 3.4)
y = pmin(t, c)
delta = as.numeric(t<=c)
test = data.frame(time=y, status=delta, x1=x1, x2=x2)

# calculate the atrributable fraction function of x1 adjusting for x2
result=paf(Surv(time, status) ~ x1 + x2, data=test, cov=c('x1'))
result$fit.cox
cbind(result$time, result$est, result$se, result$low, result$upp)[1:10, ]
# Calculate the unadjusted attributable fraciton function of x1
result=paf(Surv(time, status) ~ x1, data=test, cov=c('x1'))
```

`plot.paf` *Plot method for* `paf` *objects*

Description

Plot the attributable fraction function obtained by the `paf` function.

Usage

```
## S3 method for class 'paf'
plot(x, conf.int = TRUE, lty = 1, col = 1, ylim =
NULL, xlab = "Time",
ylab = "Attributable Fraction Function", ...)
```

Arguments

`x`	an object of class `paf` which is retured by the `paf` function.
`conf.int`	determines whether confidence intervals will be plotted. The default is TRUE.
`lty`	an integer specifying line type.
`col`	an integer specifying color type.
`ylim`	a vector specifying the lower and upper boundaries for y values.
`xlab`	label given to the x-axis with "Time" as default.
`ylab`	label given to the y-axis with "Attributable Fraction Function" as default.
`...`	other arguments allowed for the general `plot` function.

Author(s)

Li Chen

References

Chen L, Lin DY, Zeng D. (2010). Attributable fraction functions for censored event times. *Biometrika* **97**, 713-726.

See Also

par, paf.

Examples

```
# simulated data set from a Cox model
n = 1000
x1 = as.numeric(runif(n)>0.5)
x2 = x1 + rnorm(n)
t = exp(-x1 - 0.5 * x2) * rexp(n, rate = 0.1)
c = runif(n, 0, 3.4)
y = pmin(t, c)
delta = as.numeric(t<=c)
test = data.frame(time=y, status=delta, x1=x1, x2=x2)

# calculate the atrributable fraction function of x1 adjusting for x2
result=paf(Surv(time, status) ~ x1 + x2, data=test, cov=c('x1'))
# plot the attributable fraction function
plot(result)
```

Index