Package 'LYG'

July 24, 2023

Title LYG

Version 0.0.3

Date 2023-07-21

Author Li Cheung, Hormuzd Katki, Lingxiao Wang

Depends R (>= 3.5.0), survey, data.table, matrixStats

Description Estimating life-years gained from prevention services.

Maintainer William Wheeler <WheelerB@imsweb.com>

License GPL-2

NeedsCompilation no

R topics documented:

`LYG-package` *Life Years Gained*

Description

To estimate individualized life-years gained from prevention services.

Details

The current bedrock of precision prevention is selecting high-risk individuals for screening or other prevention services. However, those at highest risk do not necessarily have highest benefit from prevention services, if those at highest risk tend to be elderly or have comorbidities that substantially reduce their life-years gained from the service. This package computes individuals' expected gain in life-years from prevention services which can be used as an alternative to using individualized risk as the selection criterion. The expected life-years gained are estimated from combining trial and survey data.

Author(s)

Li Cheung <li.cheung@nih.gov>, Hormuzd Katki, Lingxiao Wang

References

Wang, L., Katki, H.A., Chaturvedi, A.K., Cheung, L.C. (2023). Moving from individualized risk-based prevention to benefit-based prevention: Estimating individualized life-years gained from prevention services as a basis for eligibility.

`data` *Data for examples*

Description

Data for `lyg` examples.

Details

Data frames for the trial data (samp.t) and for the survey data (samp.s).

See Also

`lyg`

Examples

```
data(data, package="LYG")

# Display a few rows of the data sets
samp.s[1:5, ]
samp.t[1:5, ]
```

`lyg` *Life Years Gained*

Description

Estimates individualized life years gained from prevention services

Usage

```
lyg(fit.survey, fit2.survey, fit.trial, fit2.trial=NULL,
                time.eff.end=NULL, time.cutoff=NULL,
                x0=NULL, calc.var=TRUE, save.memory=FALSE)
```

Arguments

`fit.survey`	Return object from coxph or svycoxph for the major event of interest in the survey data. NOTE: If this object is returned from coxph, then the data frame used for the fit must still be in memory.
`fit2.survey`	Return object from coxph or svycoxph for the competing event in the survey data. NOTE: If this object is returned from coxph, then the data frame used for the fit must still be in memory.
`fit.trial`	Return object from coxph or svycoxph for the major event of interest in the trial data. NOTE: If this object is returned from coxph, then the data frame used for the fit must still be in memory.
`fit2.trial`	NULL or return object from coxph or svycoxph for the competing event in the trial data. NOTE: If this object is returned from coxph, then the data frame used for the fit must still be in memory.
`time.eff.end`	The time when the intervention effect on reducing the risk of the major event becomes zero. The default is the maximum event time.
`time.cutoff`	The cutoff time for life years gained estimation. The default is the maximum event time.
`x0`	Matrix or data frame of covariate values to computes estimates for. This object must have column names as the names of the parameter estimates in `fit.survey`. The default is `model.matrix(fit.survey)`.
`calc.var`	TRUE or FALSE to compute variance estimates. The default is TRUE.
`save.memory`	TRUE or FALSE to save memory when the estimates are computed. Setting this option to TRUE will cause the program to run slower but save a substantial amount of memory. See details. The default is FALSE.

Details

See the manuscript for complete details of the methods.

When `save.memory = TRUE`, the program will loop over the distinct covariate profiles in the data `x0` in order to reduce the memory needed. One way to make the program run faster when `save.memory = TRUE`, is to use categorical covariates in `fit.survey`.

Value

A list containing the estimated life years gained and their variances.

Examples

```
data(data, package="LYG")

# Fit Cox regression models to the trial data
cox_t.d1 <- coxph(Surv(t_t, d1_t) ~ y, data=samp.t, robust=TRUE, ties="breslow")
cox_t.d2 <- coxph(Surv(t_t, d2_t) ~ y, data=samp.t, robust=TRUE, ties="breslow")

# Example 1
# Fit Cox regression models for the outcome and other predictors to
#  another data source with the complex sample design considered
ds.s <- svydesign(ids=~PSU, strata = ~strata, weights=~wt,
                  data=samp.s, nest = TRUE)
svycox_s.d1 <- svycoxph(Surv(t_s, d1_s) ~ x1 + x2, ds.s)
svycox_s.d2 <- svycoxph(Surv(t_s, d2_s) ~ x1 + x2, ds.s)
```

```
svy.x0=data.frame(x1 = c(svyquantile(~x1, quantiles=c(.10, .25, .5, .75, .90), design = ds.s)$x1[, 1]),
            x2 = c(svyquantile(~x2, quantiles=c(.10, .25, .5, .75, .90), design = ds.s)$x2[, 1]))
names(svy.x0) # should match with the names of covariates in model cox_s.d1 and cox_s.d2.


ret <- lyg(fit.survey = svycox_s.d1, fit2.survey = svycox_s.d2,
           fit.trial = cox_t.d1, fit2.trial=cox_t.d2, x0 = svy.x0,
           save.memory=FALSE)


# Example 2
# Fit Cox regression models for the outcome and other predictors to another
#   data source if the sample design is unknown
cox_s.d1 <- coxph(Surv(t_s, d1_s) ~ x1 + x2, data = samp.s, robust=TRUE, ties = "breslow")
cox_s.d2 <- coxph(Surv(t_s, d2_s) ~ x1 + x2, data = samp.s, robust=TRUE, ties = "breslow")

x0=data.frame(x1 = quantile(samp.s$x1, prob=c(.10, .25, .5, .75, .90)),
              x2 = quantile(samp.s$x2, prob=c(.10, .25, .5, .75, .90)))

names(x0) # should match with the names of covariates in model cox_s.d1 and cox_s.d2.


ret <- lyg(fit.survey = cox_s.d1, fit2.survey = cox_s.d2,
           fit.trial = cox_t.d1, fit2.trial=cox_t.d2, x0 = x0,
           save.memory=FALSE)
```

Index