Package 'RECSO'

July 2, 2014

Type Package

Title Robust and Efficient Analysis using Control function approach,of a Secondary Outcome

Version 1.0

Date 2013-08-21

Author Tamar Sofer

Maintainer Tamar Sofer <tsofer@hsph.harvard.edu>

Description

Performs robust and efficient analysis of the effect of exposure on a secondary outcome in a case-control study.

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-11-27 18:06:08

R topics documented:

RECSO-package *Robust and Efficient analysis using Control function, of Secondary Outcomes.*

Description

Performs robust and efficient analysis of the effect of exposure on a secondary outcome in a case-control study.

Details

Package:	RECSO
Type:	Package
Version:	1.0
Date:	2013-08-21
License:	What license is it under?

The main function in this package is secondary.outcome.effect. It calculates the effect in the population of exposure covariates on a secondary outcome in a case-control study. It uses the disease status, prevalence of disease in the population. It either models the probability of disease given covariates using logistic regression, or accepts estimated probabilities for each subjects. It also models the "selection bias" function resulting from the case-control sampling.

Author(s)

Tamar Sofer

Maintainer: Tamar Sofer <tamar.sofer@post.harvard.edu>

References

Sofer, T. and Tchetgen Tchetgen EJ. "Control function assisted IPW estimation with a secondary outcome in case-control studies" (In preparation).

Examples

```
### Identity link example:



n.case <- 500
n.cont <- 500
n <- n.case + n.cont
D <- c(rep(1, n.case), rep(0, n.cont))
X1 <- c(rnorm(n.case, 0,4), rnorm(n.cont, 2,4))
X2 <- rnorm(n, 2,2)
```

```
p.D.pop <- 0.12
p.D.cc.samp <- n.case/(n.case + n.cont)
pi.X <- fitted(glm(D ~ X1, family = "binomial"))
p.DX <- expit( logit(pi.X) +
log(p.D.pop*(1-p.D.cc.samp)/(p.D.cc.samp*(1-p.D.pop))) )

mean.Y.DX <- 50 + 3*X1 - (D - p.DX)*(3 + 2*X1 - 2*X1*X2)
Y <- mean.Y.DX + rnorm(n, 0,4)
X <- data.frame(X1, X2)
X$X1X2 <- X$X1*X$X2

## misspecified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X2"),
p.D.pop = 0.12, b.init.type = "ipw", link = "identity")

## correctly specified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X1", "X2", "X1X2"),
p.D.pop = 0.12, b.init.type = "ipw", link = "identity")


### log link example:
beta <- c(3, 0.7, 0.4, 0.4)
alpha <- c(0.5, 0.3, 0.3,0)
p.D.pop <- 0.12
n.case <- 500
n.cont <- 500
n <- n.case + n.cont
D <- c(rep(1, n.case), rep(0, n.cont))
X1 <- c(rnorm(n.case, 1, 0.2), rnorm(n.cont, 1.5, 0.2))
X2 <- rnorm(n,1, 0.2)
p.D.cc.samp <- n.case/(n.case + n.cont)
pi.X <- fitted(glm(D ~ X1, family = "binomial"))
p.DX <- expit( logit(pi.X) +
log(p.D.pop*(1-p.D.cc.samp)/(p.D.cc.samp*(1-p.D.pop))) )

X <- cbind(1,X1, X2, X1*X2)
colnames(X) <- c("intercept", "X1", "X2", "X1X2")
mean.Y.DX <- exp(X %*% beta +  D*(X %*% alpha) - log(exp(X %*% alpha)*p.DX + 1-p.DX) )
Y <- rpois(n, mean.Y.DX)

## misspecified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2", "X1X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X2"),
p.D.pop = p.D.pop, b.init.type = "ipw", link = "log")

## correctly specified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2", "X1X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X1", "X2", "X1X2"),
p.D.pop = p.D.pop, b.init.type = "ipw", link = "log")
```

calc.b.init *Starting valued of the main effect parameter*

Description

Calculates a starting point for parameter estimates. This function is called by the main function "secondary.outcome.effect".

Usage

```
calc.b.init(Y, design.mat.main.model, weights, b.init.type = "ipw", link = "identity")
```

Arguments

Y	Secondary outcome of interest.
design.mat.main.model	Design matrix for the effect of exposure (and covariates) on the secondary outcome
weights	Inverse probability weights (probability of being in the sample given disease status)
b.init.type	type of initial regression parameter estimator. Could be "ipw" (default), "pooled" (based on all observations, ignoring case-control sampling) or "controls" (estimated from controls only data).
link	Type of link functions. Either "identity" or "log".

Value

Initial estimator of the effect of covariates on the secondary outcome.

expit *expit function*

Description

Performs the expit transformation

Usage

```
expit(x)
```

Arguments

x	a numeric matrix/vector/scalar

Value

each value of x is return as expit of the original value

`logit` *logit function*

Description

Performs the logit transformation

Usage

```
logit(x)
```

Arguments

`x`	a numeric matrix/vector/scalar

Value

each value of x is return as logit of the original value

`secondary.outcome.effect` *The effect of exposure on secondary outcome*

Description

The main function of the package.

Usage

```
secondary.outcome.effect(Y, X, D, main.effect.vars, disease.vars,
selection.bias.function.vars, p.D.pop, b.init.type = "ipw",
link = "identity", p.Dx.cc = NULL, max.iter = 500, eps = 1e-06)
```

Arguments

`Y`	Vector of the secondary outcome of interest.
`X`	Matrix of all covariates and exposure variables needed (could have more variables than actually used).
`D`	Vector of disease status
`main.effect.vars`	Exposure/covariates to estimate their effect on the secondary outcome, in the general populations. Must be column names of X.

disease.vars	Exposure/covariates to estimate their effect on the disease, in the general/case-control study populations. Must be column names of X.
selection.bias.function.vars	Exposure/covariates to estimate their effect on the selection bias, in the general populations. Must be column names of X.
p.D.pop	Disease prevalence in the general population
b.init.type	Type of initial estimator for the main effect? Could be "ipw" (default), "pooled" (based on all observations, ignoring case-control sampling) or "controls" (estimated from controls only data).
link	Type of link functions. Either "identity" or "log".
p.Dx.cc	Probability of disease given covariates, in the case control study, for each subject. If not given, this is modeled in a logistic regression.
max.iter	Maximum number of iterations of the Newton-Raphson algorithm. Default is 500.
eps	Criterion for convergence of the estimated vector. (Maximum absolute value of the difference between every two entries of the vector).

Value

beta.hat	Vector giving the estimated coefficients of the effect of main.effect.vars (and intercept) on Y.
cov.beta.hat	Estimated covariance matrix of the beta.hat variables.
alpha.hat	Vector giving the estimated coefficients of the effect of selection.bias.function.vars (and intercept) on the selection bias function.
cov.alpha.hat	Estimated covariance matrix of the alpha.hat variables.

Author(s)

Tamar Sofer

References

Sofer, T., Tchetgen Tchetgen, EJ. “Control function assisted IPW estimation with a secondary outcome in case-control studies" (In preparation).

Examples

```
### Identity link example:


n.case <- 500
n.cont <- 500
n <- n.case + n.cont
D <- c(rep(1, n.case), rep(0, n.cont))
X1 <- c(rnorm(n.case, 0,4), rnorm(n.cont, 2,4))
X2 <- rnorm(n, 2,2)
```

```
p.D.pop <- 0.12
p.D.cc.samp <- n.case/(n.case + n.cont)
pi.X <- fitted(glm(D ~ X1, family = "binomial"))
p.DX <- expit( logit(pi.X) +
log(p.D.pop*(1-p.D.cc.samp)/(p.D.cc.samp*(1-p.D.pop))) )

mean.Y.DX <- 50 + 3*X1 - (D - p.DX)*(3 + 2*X1 - 2*X1*X2)
Y <- mean.Y.DX + rnorm(n, 0,4)
X <- data.frame(X1, X2)
X$X1X2 <- X$X1*X$X2

## misspecified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X2"),
p.D.pop = 0.12, b.init.type = "ipw", link = "identity")

## correctly specified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X1", "X2", "X1X2"),
p.D.pop = 0.12, b.init.type = "ipw", link = "identity")


### log link example:
beta <- c(3, 0.7, 0.4, 0.4)
alpha <- c(0.5, 0.3, 0.3,0)
p.D.pop <- 0.12
n.case <- 500
n.cont <- 500
n <- n.case + n.cont
D <- c(rep(1, n.case), rep(0, n.cont))
X1 <- c(rnorm(n.case, 1, 0.2), rnorm(n.cont, 1.5, 0.2))
X2 <- rnorm(n,1, 0.2)
p.D.cc.samp <- n.case/(n.case + n.cont)
pi.X <- fitted(glm(D ~ X1, family = "binomial"))
p.DX <- expit( logit(pi.X) +
log(p.D.pop*(1-p.D.cc.samp)/(p.D.cc.samp*(1-p.D.pop))) )

X <- cbind(1,X1, X2, X1*X2)
colnames(X) <- c("intercept", "X1", "X2", "X1X2")
mean.Y.DX <- exp(X %*% beta +  D*(X %*% alpha) - log(exp(X %*% alpha)*p.DX + 1-p.DX) )
Y <- rpois(n, mean.Y.DX)

## misspecified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2", "X1X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X2"),
p.D.pop = p.D.pop, b.init.type = "ipw", link = "log")

## correctly specified model for selection bias function:
result <- secondary.outcome.effect(Y, X, D,  main.effect.vars = c("X1", "X2", "X1X2"),
disease.vars = c("X1", "X2"), selection.bias.function.vars = c("X1", "X2", "X1X2"),
p.D.pop = p.D.pop, b.init.type = "ipw", link = "log")
```

`secondary.outcome.effect.identity` *The effect of exposure on secondary outcome, modeled via the identity link*

Description

This function is called by the main function secondary.outcome.effect.

Usage

```
secondary.outcome.effect.identity(Y, design.mat.main.model, b.0, D,
design.mat.bias.model, weights, p.Dx.pop, p.Dx.cc, max.iter = 500, eps = 1e-06)
```

Arguments

`Y`	Vector of the secondary outcome of interest.
`design.mat.main.model`	Design matrix for the effect of exposure (and covariates) on the secondary outcome
`b.0`	Starting value for main effect estimates
`D`	Vector of disease status
`design.mat.bias.model`	Design matrix for the effect of covariates on the function modeling the bias resulting from case-control sampling
`weights`	Inverse probability weights (probability of being in the sample given disease status)
`p.Dx.pop`	Estimated probability of disease given covariates in the population (calculated by secondary.outcome.effect)
`p.Dx.cc`	Estimated probability of disease given covariates in the case-control study (calculated by secondary.outcome.effect)
`max.iter`	Maximum number of iterations of the Newton-Raphson algorithm. Default is 500.
`eps`	Criterion for convergence of the estimated vector. (Maximum absolute value of the difference between every two entries of the vector).

secondary.outcome.effect.log
The effect of exposure on secondary outcome, modeled via the log link

Description

This function is called by the main function secondary.outcome.effect.

Usage

```
secondary.outcome.effect.log(Y, design.mat.main.model, b.0, D,
design.mat.bias.model, weights, p.Dx.pop, p.Dx.cc, max.iter = 500, eps = 1e-06)
```

Arguments

Y	Vector of the secondary outcome of interest.
design.mat.main.model	Design matrix for the effect of exposure (and covariates) on the secondary outcome
b.0	Starting value for main effect estimates
D	Vector of disease status
design.mat.bias.model	Design matrix for the effect of covariates on the function modeling the bias resulting from case-control sampling
weights	Inverse probability weights (probability of being in the sample given disease status)
p.Dx.pop	Estimated probability of disease given covariates in the population (calculated by secondary.outcome.effect)
p.Dx.cc	Estimated probability of disease given covariates in the case-control study (calculated by secondary.outcome.effect)
max.iter	Maximum number of iterations of the Newton-Raphson algorithm. Default is 500.
eps	Criterion for convergence of the estimated vector. (Maximum absolute value of the difference between every two entries of the vector).

Index