

# Package ‘sptm’

October 10, 2014

**LazyLoad** yes

**LazyData** yes

**Version** 14.10-8

**Date** 14.10-08

**Title** SemiParametric Transformation Model Methods

**Author** Youyi Fong <yfong@fhcrc.org>, Krisztian Sebestyen <ksebestyen@gmail.com>

**Maintainer** Youyi Fong <yfong@fhcrc.org>

**Depends** R (>= 3.0.0), survival, survey

**Imports** kyotil

**Suggests** RUnit, mvtnorm, Matrix, MASS

**Description** Semiparametric transformation model, calibration weights

**License** GPL (>= 2)

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2014-10-10 07:27:16

## R topics documented:

enhanced.ipw.coxph . . . . .	2
rstm . . . . .	2
sim.fong . . . . .	3
sim.kong . . . . .	5
stm . . . . .	5

<b>Index</b>	<b>7</b>
--------------	----------

---

enhanced.ipw.coxph      *Enhanced Inverse Probability Weighted coxph*

---

**Description**

enhanced.ipw.coxph is a wrapper function for calling svycoxph of survey package.

**Usage**

```
enhanced.ipw.coxph (formula, dat, strata.formula, subset, imputation.formulae,  
                  verbose=FALSE)
```

**Arguments**

formula	a formula that gives the model we are interested to fit
dat	a data frame
strata.formula	a formula that gives how two phase sampling is done
subset	a vector of logicals that give which observations are included in phase 2
imputation.formulae	a list of formulae or a single formula that give models to impute missing data
verbose	Boolean

**Value**

An object of class svycoxph.

**Author(s)**

Youyi Fong <yfong@fhcrc.org>

---

rstm      *Simulate failure time from a semiparametric transformation model*

---

**Description**

Simulate failure time from a semiparametric transformation model

**Usage**

```
rstm(n, family = c("PH", "P0", "P2"), linear.predictors, baseline.hazard = 1)
```

**Arguments**

n integer. Sample size  
 family string.  
 linear.predictors vector. It can also be a matrix of 1 column, the dimension will be dropped  
 baseline.hazard numeric.

**Details**

Called by sim.fong

---

sim.fong *Data Simulation as in Fong and Gilbert (2014)*

---

**Description**

Simulate data as in Fong and Gilbert (2014).

**Usage**

```
sim.fong (n, family=c("PH","PO","P2"), beta,
  random.censoring=c("0%","20%","60%"), prevalence=0.1, non.adherence.ratio=0,
  design=c("FULL","CC"), auxiliary=c("weak","good","excellent","none"),
  seed=NULL, var.S=1, var.W=1)
```

**Arguments**

n integer. Sample size  
 family string. Link functions in the semiparametric transformation model  
 beta numerical vector. Coefficients of the linear model  
 random.censoring string. Random censoring in addition to administrative censoring  
 prevalence numerical. Proportion of cases among  $z=0$  when there is no random censoring and non-adherence ratio is 0  
 design string. Full cohort or case-cohort (finite population sampling)  
 auxiliary string.  
 seed integer. Random generator seed  
 var.S numeric. Variance of the phase II covariate s  
 var.W numeric. Variance of the baseline covariate w  
 non.adherence.ratio ratio of non-adherent

**Details**

The number of rows is the size of the full cohort. Adherence ratio works as a Bernoulli variable. Prevalence is used to compute baseline hazard function based on some empirical evidence.

**Value**

If design is FULL, returns a data frame of:

ft	failure time
C	censoring time
X	smaller of the ft and C
d	event indicator
z	baseline covariate z
s	phase II covariate s

If design is CC, returns a data frame of:

ft	failure time
C	censoring time
X	smaller of the ft and C
d	event indicator
z	baseline covariate z
s	phase II covariate s
w	baseline auxiliary covariate w

**Examples**

```
dat = sim.fong(n=10000, family="PH", beta=c(log(.5), log(.7), log(1.2)), design="CC",
  auxiliary="weak", seed=1, prevalence=0.1, non.adherence.ratio=0, random.censoring="0")
mean(dat$d[dat$z==0])
```

```
dat = sim.fong(n=10000, family="PH", beta=c(log(.5), log(.7), log(1.2)), design="CC",
  auxiliary="weak", seed=1, prevalence=0.1, non.adherence.ratio=0.15, random.censoring="0")
sum(dat$d & !is.na(dat$s))
sum(!dat$d & !is.na(dat$s)) / sum(dat$d & !is.na(dat$s))
```

```
dat = sim.fong(n=10000, family="PH", beta=c(log(.5), log(.7), log(1.2)), design="CC",
  auxiliary="weak", seed=1, prevalence=0.1, non.adherence.ratio=0.15, random.censoring="20")
sum(dat$d & !is.na(dat$s))
sum(!dat$d & !is.na(dat$s)) / sum(dat$d & !is.na(dat$s))
```

---

 sim.kong

*Data Simulation as in Kong et al. (2004)*


---

**Description**

Simulate data as in Kong et al. (2004).

**Usage**

```
sim.kong(gamma, beta, design = "FULL", rho = 0.9, seed = 1, impute = FALSE, ppi)
```

**Arguments**

gamma  
 beta  
 design  
 rho  
 seed  
 impute  
 ppi

---

 stm

*Fit a semiparametric transformation model*


---

**Description**

Fit a semiparametric transformation model

**Usage**

```
stm (formula, dat, strata.formula, phase2.ind=NULL, imputation.formula=NULL,
     family=c("PH", "PO", "P2"), ee=c("fine2", "fine1", "kong"), var.est.type=c("1", "2"),
     t0, init=NULL, maxit=1000,
     intermediate=FALSE, verbose=FALSE, show.time.elapsed=TRUE)
```

```
## S3 method for class 'stm'
getFixedEf(object, ...)
```

**Arguments**

<code>formula</code>	formula. Regression model of interest
<code>dat</code>	data frame.
<code>strata.formula</code>	formula.
<code>phase2.ind</code>	Boolean vector. If TRUE, phase II samples; if FALSE, phase I samples. If NULL, will try to infer from which subjects have phase II variables. Should not be 0/1
<code>imputation.formula</code>	formula. If not NULL, calibration weighting is done
<code>family</code>	string.
<code>ee</code>	string. Type of design matrix used in estimating equation
<code>var.est.type</code>	string. 1: one-stage estimator, 2: two-stage estimator
<code>t0</code>	numeric. Should be close to the end of study time
<code>init</code>	numerical vector.
<code>maxit</code>	integer. Maximum number of iterations in the optimization process
<code>intermediate</code>	Boolean.
<code>verbose</code>	Boolean.
<code>show.time.elapsed</code>	Boolean.
<code>object</code>	an object of type stm
<code>...</code>	additional arguments

**Details**

Fit `stm` both with and without calibration. Calls `stm.internal`.

**Value**

An object of type `stm`

**Examples**

```
n=100
beta= c(log(.5), log(.7), log(1.2))
t0=2.9999
init = c(log(0.0373*t0),beta)
dat = sim.fong(n, family="PH", beta, random.censoring="0", design="CC", auxiliary="weak", seed=1)

est = stm(formula=Surv(X,d) ~ z + s + z:s, dat, strata.formula=~d, family="PH", t0=t0, init=init,
var.est.type="1", verbose=3)
```

# Index

[enhanced.ipw.coxph](#), [2](#)

[getFixedEf.stm\(stm\)](#), [5](#)

[rstm](#), [2](#)

[sim.fong](#), [3](#)

[sim.kong](#), [5](#)

[stm](#), [5](#)