Package ‘twoStageDesignTMLE’

July 22, 2025

Title Targeted Maximum Likelihood Estimation for Two-Stage Study Design

Version 1.0.1.2

Author Susan Gruber [aut, cre],
Mark van der Laan [aut]

Maintainer Susan Gruber `<sgruber@TLrevolution.com>`

Description An inverse probability of censoring weighted (IPCW) targeted maximum likelihood estimator (TMLE) for evaluating a marginal point treatment effect from data where some variables were collected on only a subset of participants using a two-stage design (or marginal mean outcome for a single arm study). A TMLE for conditional parameters defined by a marginal structural model (MSM) is also available.

Depends tmle (>= 2.0)

Suggests dbarts (>= 0.9-18), glmnet

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Repository CRAN

Date/Publication 2025-02-05 16:00:02 UTC

Contents

`estimatePi` *estimatePi*

Description

Typically not called directly by the user. Function for modeling the two-stage missingness mechanism and evaluating conditional probabilities for each observation

Usage

```
estimatePi(
  Y,
  A,
  W,
  condSetNames,
  W.Q,
  Delta.W,
  V.msm = NULL,
  piform,
  pi.SL.library,
  id,
  V,
  discreteSL,
  verbose,
  pi = NULL,
  obsWeights = rep(1, nrow(W))
)
```

Arguments

`Y`	outcome
`A`	binary treatment indicator
`W`	covariate matrix observed on everyone
`condSetNames`	Variables to include as predictors of missingness in `W.stage2`, any combination of `Y`, `A`, and either `W` (for all covariates in `W`) or individual covariate names in `W`
`W.Q`	additional covariates based on preliminary outcome regression
`Delta.W`	binary indicator of missing second stage covariates
`V.msm`	optional additional covariates to condition on beyond `W`
`piform`	parametric regression formula for estimating `pi`
`pi.SL.library`	super learner library for estimating `pi`
`id`	Identifier of independent units of observation, e.g., clusters
`V`	number of cross validation folds for estimating `pi` using super learner
`discreteSL`	Use discrete super learning when `TRUE`, otherwise ensemble super learning

`verbose`	When `TRUE` prints informational messages
`pi`	optional vector of user-specified probabilities
`obsWeights`	optional weights for evaluating pi

Value

list containing the predicted probabilities, estimation method coefficients in parametric regression model (if piform supplied), indicator of whether discrete or ensemble SL was used.

`evalAugW` *.evalAugW calls TMLE to use super learner to evalute preliminary predictions for Q(0,W) and Q(1,W) conditioning on stage 1 covariates*

Description

.evalAugW calls TMLE to use super learner to evalute preliminary predictions for Q(0,W) and Q(1,W) conditioning on stage 1 covariates

Usage

```
evalAugW(Y, A, W, Delta, id, family, SL.library)
```

Arguments

`Y`	outcome vector
`A`	binary treatment indicator
`W`	covariate matrix
`Delta`	outcome missingness indicator
`id`	identifier of i.i.d. unit
`family`	outcome regression family
`SL.library`	super learner library for outcome regression modeling

Value

`W.Q`, nx2 matrix of outcome predictions based on stage 1 covariates

---

`print.summary.twoStageTMLE`

print.summary.twoStageTMLE

Description

print.summary.twoStageTMLE

Usage

```
## S3 method for class 'summary.twoStageTMLE'
print(x, ...)
```

Arguments

`x`	an object of class summary.twoStageTMLE
`...`	additional arguments (i)

Value

print object

`print.twoStageTMLE`

print.twoStageTMLE

Description

print.twoStageTMLE

Usage

```
## S3 method for class 'twoStageTMLE'
print(x, ...)
```

Arguments

`x`	an object of class twoStageTMLE
`...`	additional arguments (i)

Value

print tmle results using print.tmle method from tmle package

`setV` *Utilities setV Set the number of cross-validation folds as a function of effective sample size See Phillips 2023 doi.org/10.1093/ije/dyad023*

Description

Utilities setV Set the number of cross-validation folds as a function of effective sample size See Phillips 2023 doi.org/10.1093/ije/dyad023

Usage

```
setV(n.effective)
```

Arguments

`n.effective`	the effective sample size

Value

the number of cross-validation folds

`summary.twoStage` *summary.twoStageTMLE*

Description

Summarizes estimation procedure for missing 2nd stage covariates

Usage

```
## S3 method for class 'twoStage'
summary(object, ...)
```

Arguments

`object`	An object of class `twoStageTMLE`
`...`	Other arguments passed to the tmle function in the tmle package

Value

A list containing the missingness model, terms, coefficients, type,

---

`summary.twoStageTMLE` *summary.twoStageTMLE*

Description

summary.twoStageTMLE

Usage

```
## S3 method for class 'twoStageTMLE'
summary(object, ...)
```

Arguments

`object`	an object of class twoStageTMLE
`...`	additional arguments (ignored)

Value

list summarizing the two-stage procedure components, summary of the twoStage missingness estimation summary of the tmle for estimating the parameter

---

`twoStageDesignTMLENews` *twoStageDesignTMLENews Get news about recent updates and bug fixes*

Description

twoStageDesignTMLENews Get news about recent updates and bug fixes

Usage

```
twoStageDesignTMLENews(...)
```

Arguments

`...`	ignored

Value

invisible character string giving the path to the file found.

`twoStageTMLE` *twoStageTMLE*

Description

Inverse probability of censoring weighted TMLE for evaluating parameters when the full set of covariates is available on only a subset of observations.

Usage

```
twoStageTMLE(
  Y,
  A,
  W,
  Delta.W,
  W.stage2,
  Z = NULL,
  Delta = rep(1, length(Y)),
  pi = NULL,
  piform = NULL,
  pi.SL.library = c("SL.glm", "SL.gam", "SL.glmnet", "tmle.SL.dbarts.k.5"),
  V.pi = 10,
  pi.discreteSL = TRUE,
  condSetNames = c("A", "W", "Y"),
  id = NULL,
  Q.family = "gaussian",
  augmentW = TRUE,
  augW.SL.library = c("SL.glm", "SL.glmnet", "tmle.SL.dbarts2"),
  rareOutcome = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`Y`	outcome
`A`	binary treatment indicator
`W`	covariate matrix observed on everyone
`Delta.W`	binary indicator of missing second stage covariates
`W.stage2`	matrix of second stage covariates observed on subset of observations
`Z`	optional mediator of treatment effect for evaluating a controlled direct effect
`Delta`	binary indicator of missing value for outcome `Y`
`pi`	optional vector of missingness probabilities for `W.stage2`
`piform`	parametric regression formula for estimating `pi` (see Details)

`pi.SL.library`	super learner library for estimating `pi` (see Details)
`V.pi`	number of cross validation folds for estimating `pi` using super learner
`pi.discreteSL`	Use discrete super learning when `TRUE`, otherwise ensemble super learning
`condSetNames`	Variables to include as predictors of missingness in `W.stage2`, any combination of `Y`, `A`, and either `W` (for all covariates in `W`), or individual covariate names in `W`
`id`	Identifier of independent units of observation, e.g., clusters
`Q.family`	Regression family for the outcome
`augmentW`	When `TRUE` include predicted values for the outcome the set of covariates used to model the propensity score
`augW.SL.library`	super learner library for preliminary outcome regression model (ignored when `augmentW` is `FALSE`)
`rareOutcome`	When `TRUE` specifies less ambitious SL for Q in call to `tmle` (discreteSL, glm, glmnet, bart library, `V=20`)
`verbose`	When `TRUE` prints informational messages
`...`	other parameters passed to the tmle function (not checked)

Details

When using `piform` to specify a parametric model for pi that conditions on the outcome use `Delta.W` as the dependent variable and `Y.orig` on the right hand side of the formula instead of `Y`. When writing a user-defined SL wrapper for inclusion in `pi.SL.library` use `Y` on the left hand side of the formula. If specific covariate names are used on the right hand side use `Y.orig` to condition on the outcome.

Value

object of class 'twoStageTMLE'.

`tmle`	Treatment effect estimates and summary information
`twoStage`	IPCW weight estimation summary, `pi` are the probabilities, `coef` are SL weights or coefficients from glm fit, `type` of estimation procedure, `discreteSL` flag indicating whether discrete super learning was used
`augW`	Matrix of predicted outcomes based on stage 1 covariates only

See Also

- `tmle::tmle()` for details on customizing the estimation procedure
- `twoStageTMLEmsm()` for estimating conditional effects
- S Rose and MJ van der Laan. A Targeted Maximum Likelihood Estimator for Two-Stage Designs. *Int J Biostat.* 2011 Jan 1; 7(1): 17. doi:10.2202/15574679.1217

Examples

```
n <- 1000
W1 <- rnorm(n)
W2 <- rnorm(n)
W3 <- rnorm(n)
A <- rbinom(n, 1, plogis(-1 + .2*W1 + .3*W2 + .1*W3))
Y <- 10 + A + W1 + W2 + A*W1 + W3 + rnorm(n)
d <- data.frame(Y, A, W1, W2, W3)
# Set 400 with data on W3, more likely if W1 > 1
n.sample <- 400
p.sample <- 0.5 + .2*(W1 > 1)
rows.sample <- sample(1:n, size = n.sample, p = p.sample)
Delta.W <- rep(0,n)
Delta.W[rows.sample] <- 1
W3.stage2 <- cbind(W3 = W3[Delta.W==1])
#1. specify parametric models and do not augment W (fast, but not recommended)
result1 <- twoStageTMLE(Y=Y, A=A, W=cbind(W1, W2), Delta.W = Delta.W,
   W.stage2 = W3.stage2, piform = "Delta.W~ I(W1 > 0) + Y.orig", V.pi= 5,
   verbose = TRUE, Qform = "Y~A+W1",gform="A~W1 + W2 +W3", augmentW = FALSE)
summary(result1)

#2. specify a parametric model for conditional missingness probabilities (pi)
#   and use default values to estimate marginal effect using \code{tmle}
result2 <- twoStageTMLE(Y=Y, A=A, W=cbind(W1, W2), Delta.W = Delta.W,
     W.stage2 = cbind(W3)[Delta.W == 1], piform = "Delta.W~ I(W1 > 0)",
     V.pi= 5,verbose = TRUE)
result2
```

`twoStageTMLEmsm` *twoStageTMLEmsm*

Description

Inverse probability of censoring weighted TMLE for evaluating MSM parameters when the full set of covariates is available on only a subset of observations, as in a 2-stage design.

Usage

```
twoStageTMLEmsm(
  Y,
  A,
  W,
  V,
  Delta.W,
  W.stage2,
  Delta = rep(1, length(Y)),
  pi = NULL,
  piform = NULL,
```

```
  pi.SL.library = c("SL.glm", "SL.gam", "SL.glmnet", "tmle.SL.dbarts.k.5"),
  V.pi = 10,
  pi.discreteSL = TRUE,
  condSetNames = c("A", "V", "W", "Y"),
  id = NULL,
  Q.family = "gaussian",
  augmentW = TRUE,
  augW.SL.library = c("SL.glm", "SL.glmnet", "tmle.SL.dbarts2"),
  rareOutcome = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`Y`	outcome of interest (missingness allowed)
`A`	binary treatment indicator
`W`	matrix or data.frame of covariates measured on entire population
`V`	vector, matrix, or dataframe of covariates used to define MSM strata
`Delta.W`	Indicator of inclusion in subset with additional information
`W.stage2`	matrix or data.frame of covariates measured in subset population
`Delta`	binary indicator that outcome Y is observed
`pi`	optional vector of sampling probabilities
`piform`	parametric regression formula for estimating `pi` (see Details)
`pi.SL.library`	super learner library for estimating `pi` (see Details)
`V.pi`	optional number of cross-validation folds for super learning (ignored when piform or pi is provided)
`pi.discreteSL`	flag to indicate whether to use ensemble or discrete super learning (ignored when piform or pi is provided)
`condSetNames`	Variables to include as predictors of missingness in `W.stage2`, any combination of `Y`, `A`, and either `W` (for all covariates in `W`), or individual covariate names in `W`
`id`	optional indicator of independent units of observation
`Q.family`	outcome regression family, "gaussian" or "binomial"
`augmentW`	set to `TRUE` to augment `W` with predicted outcome values when `A = 0` and `A = 1`
`augW.SL.library`	super learner library for preliminary outcome regression model (ignored when `augmentW` is `FALSE`)
`rareOutcome`	when `TRUE` sets `V.Q = 20`, `Q.discreteSL = TRUE`, `Q.SL.library` includes glm, glmnet, bart
`verbose`	when `TRUE` prints informative messages
`...`	other arguments passed to the `tmleMSM` function

Details

When using `piform` to specify a parametric model for pi that conditions on the outcome use `Delta.W` as the dependent variable and `Y.orig` on the right hand side of the formula instead of `Y`. When writing a user-defined SL wrapper for inclusion in `pi.SL.library` use `Y` on the left hand side of the formula. If specific covariate names are used on the right hand side use `Y.orig` to condition on the outcome.

Value

Object of class "twoStageTMLE"

Treatment effect estimates and summary information from call to `tmleMSM` function

tmle **twoStage** IPCW weight estimation summary, `pi` are the probabilities,`coef` are SL weights or coefficients from glm fit, `type` of estimation procedure, `discreteSL` flag indicating whether discrete super learning was used

augW Matrix of predicted outcomes based on stage 1 covariates only

See Also

- `tmle::tmleMSM()` for details on customizing the estimation procedure
- `twoStageTMLE()` for estimating marginal effects

Examples

```
n <- 1000
set.seed(10)
W1 <- rnorm(n)
W2 <- rnorm(n)
W3 <- rnorm(n)
A <- rbinom(n, 1, plogis(-1 + .2*W1 + .3*W2 + .1*W3))
Y <- 10 + A + W1 + W2 + A*W1 + W3 + rnorm(n)
Y.bin <- rbinom(n, 1, plogis(-4.6 - 1.8* A + W1 + W2 -.3 *A*W1 + W3))
# Set 400 obs with data on W3, more likely if W1 > 1
n.sample <- 400
p.sample <- 0.5 + .2*(W1 > 1)
rows.sample <- sample(1:n, size = n.sample, p = p.sample)
Delta.W <- rep(0,n)
Delta.W[rows.sample] <- 1
W3.stage2 <- cbind(W3 = W3[Delta.W==1])

# 1. specify parametric models, misspecified outcome model (not recommended)
result1.MSM <- twoStageTMLEmsm(Y=Y, A=A, V= cbind(W1), W=cbind(W2),
Delta.W = Delta.W, W.stage2 = W3.stage2, augmentW = FALSE,
piform = "Delta.W~ I(W1 > 0)", MSM = "A*W1", augW.SL.library = "SL.glm",
Qform = "Y~A+W1",gform="A~W1 + W2 +W3", hAVform = "A~1", verbose=TRUE)
summary(result1.MSM)

# 2. Call again, passing in previously estimated observation weights,
# note that specifying a correct model for Q improves efficiency
result2.MSM <- twoStageTMLEmsm(Y=Y, A=A, V= cbind(W1), W=cbind(W2),
Delta.W = Delta.W, W.stage2 = W3.stage2, augmentW = FALSE,
```

```
pi = result1.MSM$twoStage$pi, MSM = "A*W1",
Qform = "Y~ A + W1 + W2 + A*W1 + W3",gform="A~W1 + W2 +W3", hAVform = "A~1")
cbind(SE.Qmis = result1.MSM$tmle$se, SE.Qcor = result2.MSM$tmle$se)


#Binary outcome, augmentW, rareOutcome
result3.MSM <- twoStageTMLEmsm(Y=Y.bin, A=A, V= cbind(W1), W=cbind(W2),
Delta.W = Delta.W, W.stage2 = W3.stage2, augmentW = TRUE,
piform = "Delta.W~ I(W1 > 0)", MSM = "A*W1", gform="A~W1 + W2 +W3",
 Q.family = "binomial", rareOutcome=TRUE)
```

Index