Package 'txshift'

September 25, 2020

Title Efficient Estimation of the Causal Effects of Stochastic Interventions

Version 0.3.4

Maintainer Nima Hejazi <nh@nimahejazi.org>

Description Efficient estimation of the population-level causal effects of stochastic interventions on a continuous-valued exposure. Both one-step and targeted minimum loss estimators are implemented for a causal parameter defined as the counterfactual mean of an outcome of interest under a stochastic intervention that may depend on the natural value of the exposure (i.e., a modified treatment policy). To accommodate settings in which two-phase sampling is employed, procedures for making use of inverse probability of censoring weights are provided to facilitate construction of inefficient and efficient one-step and targeted minimum loss estimators. The causal parameter and estimation methodology were first described by Díaz and van der Laan (2013) <doi:10.1111/j.1541-0420.2011.01685.x>). Estimation of nuisance parameters may be enhanced through the Super Learner ensemble model in 'sl3', available for download from GitHub using 'remotes::install_github(``tlverse/sl3")'.

Depends R (>= 3.2.0)

Imports stats, stringr, data.table, assertthat, mvtnorm, hal9001 (>= 0.2.6), haldensify (>= 0.0.6), lspline, ggplot2, tibble, latex2exp, Rdpack

Suggests testthat, knitr, rmarkdown, covr, future, future.apply, origami (>= 1.0.3), ranger, Rsolnp, nnls, rlang

Enhances sl3 (>= 1.3.7)

License MIT + file LICENSE

URL https://github.com/nhejazi/txshift

BugReports https://github.com/nhejazi/txshift/issues

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 7.1.1

RdMacros Rdpack

NeedsCompilation no

Author Nima Hejazi [aut, cre, cph] (<https://orcid.org/0000-0002-7127-2789>),
David Benkeser [aut] (<https://orcid.org/0000-0002-1019-8343>),
Iván Díaz [ctb] (<https://orcid.org/0000-0001-9056-2047>),
Jeremy Coyle [ctb] (<https://orcid.org/0000-0002-9874-6649>),
Mark van der Laan [ctb, ths] (<https://orcid.org/0000-0003-1432-5511>)

Repository CRAN

Date/Publication 2020-09-25 13:50:02 UTC

R topics documented:

`bound_precision` *Bound Precision*

Description

Bound Precision

Usage

```
bound_precision(vals)
```

Arguments

vals	numeric vector of values in the interval [0, 1] to be bounded within arbitrary machine precision. The most common use of this functionality is to avoid indeterminate or non-finite values after the application stats::qlogis.

Details

Bound values in the unit interval to machine precision in order to avoid numerical instability issues in downstream computation.

Value

A numeric vector of the same length as vals, where the returned values are bounded to machine precision. This is intended to avoid numerical instability issues.

---

bound_propensity *Bound Generalized Propensity Score*

Description

Bound Generalized Propensity Score

Usage

```
bound_propensity(vals)
```

Arguments

vals	numeric vector of propensity score estimate values. Note that, for this parameter, the propensity score is (conditional) density and so it ought not be bounded from above.

Details

Bound estimated values of the generalized propensity score (a conditional density) to avoid numerical instability issues arising from practical violations of the assumption of positivity.

Value

A numeric vector of the same length as vals, where the returned values are bounded such that the minimum is no lower than 1/n, for the sample size n.

confint.txshift *Confidence Intervals for Counterfactual Mean Under Stochastic Intervention*

Description

Confidence Intervals for Counterfactual Mean Under Stochastic Intervention

Usage

```
## S3 method for class 'txshift'
confint(object, parm = seq_len(object$psi), level = 0.95, ...)
```

Arguments

object	An object of class txshift, as produced by invoking the function txshift, for which a confidence interval is to be computed.
parm	A numeric vector indicating indices of object$est for which to return confidence intervals.
level	A numeric indicating the level of the confidence interval to be computed.
...	Other arguments. Not currently used.

Details

Compute confidence intervals for estimates produced by txshift.

Value

A named numeric vector containing the parameter estimate from a txshift object, alongside lower and upper Wald-style confidence intervals at a specified coverage level.

Examples

```
set.seed(429153)
n_obs <- 100
W <- replicate(2, rbinom(n_obs, 1, 0.5))
A <- rnorm(n_obs, mean = 2 * W, sd = 1)
Y <- rbinom(n_obs, 1, plogis(A + W + rnorm(n_obs, mean = 0, sd = 1)))
txout <- txshift(
  W = W, A = A, Y = Y, delta = 0.5,
  estimator = "tmle",
  g_fit_args = list(
    fit_type = "hal", n_bins = 5,
    grid_type = "equal_mass",
    lambda_seq = exp(-1:-9)
  ),
  Q_fit_args = list(
    fit_type = "glm",
```

```
    glm_formula = "Y ~ ."
  )
)
confint(txout)
```

`eif` *Compute the Shift Parameter Estimate and the Efficient Influence Function*

Description

Compute the Shift Parameter Estimate and the Efficient Influence Function

Usage

```
eif(
  Y,
  Qn,
  Hn,
  estimator = c("tmle", "onestep"),
  fluc_mod_out = NULL,
  Delta = rep(1, length(Y)),
  ipc_weights = rep(1, length(Y)),
  ipc_weights_norm = rep(1, length(Y))
)
```

Arguments

`Y`	A `numeric` vector of the observed outcomes.
`Qn`	An object providing the value of the outcome evaluated after imposing a shift in the treatment. This object is passed in after being constructed by a call to the internal function `est_Q`.
`Hn`	An object providing values of the auxiliary ("clever") covariate, constructed from the treatment mechanism and required for targeted minimum loss-based estimation. This object object should be passed in after being constructed by a call to the internal function `est_Hn`.
`estimator`	The type of estimator to be fit, either `"tmle"` for targeted maximum likelihood estimation or `"onestep"` for a one-step estimator.
`fluc_mod_out`	An object giving values of the logistic tilting model for targeted minimum loss estimation. This type of object should be the output of the internal routines to perform this step of the TML estimation procedure, as given by `fit_fluctuation`.
`Delta`	Indicator for missingness. Used for compatibility with the routine to compute IPCW-TML estimates.
`ipc_weights`	A `numeric` vector that gives inverse probability of censoring weights for each observation. These are generated by invoking the routines for estimating the censoring mechanism.

ipc_weights_norm
: A numeric vector of the weights described in the previous argument. In this case, the weights are normalized.

Details

Estimate the value of the causal parameter alongside statistical inference for the parameter estimate based on the efficient influence function of the target parameter, which takes the following form:

Value

A list containing the parameter estimate, estimated variance based on the efficient influence function (EIF), the estimate of the EIF incorporating inverse probability of censoring weights, and the estimate of the EIF without the application of such weights.

---

est_g *Estimate the Generalized Propensity Score (Treatment Mechanism)*

Description

Estimate the Generalized Propensity Score (Treatment Mechanism)

Usage

```
est_g(
  A,
  W,
  delta = 0,
  ipc_weights = rep(1, length(A)),
  fit_type = c("sl", "hal"),
  sl_learners_density = NULL,
 haldensify_args = list(n_bins = c(5, 10), grid_type = c("equal_range", "equal_mass"),
    lambda_seq = exp(seq(-1, -13, length = 300)), use_future = FALSE)
)
```

Arguments

A	A numeric vector of observed treatment values.
W	A numeric matrix of observed baseline covariate values.
delta	A numeric value identifying a shift in the observed value of the treatment under which observations are to be evaluated.
ipc_weights	A numeric vector of observation-level weights, as produced by the internal procedure to estimate the censoring mechanism estimate-ipc_weights.
fit_type	A character specifying whether to use Super Learner (from **sl3**) or the Highly Adaptive Lasso (from **hal9001**) to estimate the conditional treatment density.

`sl_learners_density`
: Object containing a set of instantiated learners from **sl3**, to be used in fitting an ensemble model.

`haldensify_args`
: A `list` of argument to be directly passed to `haldensify` when `fit_type` is set to `"hal"`. Note that this invokes the Highly Adaptive Lasso instead of Super Learner and is thus only feasible for relatively small data sets.

Details

Compute the propensity score (treatment mechanism) for the observed data, including the shift. This gives the propensity score for the observed data (at the observed A) and the shift (at A - delta).

Value

A `data.table` with four columns, containing estimates of the generalized propensity score at a downshift (g(A - delta | W)), no shift (g(A | W)), an upshift (g(A + delta) | W), and an upshift of magitudie two (g(A + 2 delta) | W).

`est_Hn` *Estimate Auxiliary Covariate from Efficient Influence Function*

Description

Estimate Auxiliary Covariate from Efficient Influence Function

Usage

```
est_Hn(gn)
```

Arguments

`gn`
: An estimate of the treatment probability (propensity score), using the output provided by internal function `estimate-propensity_score`.

Details

Compute an estimate of the auxiliary covariate required to update initial estimates via logistic tilting models in targeted minimum loss estimation.

Value

A `data.table` with two columns, containing estimates of the auxiliary covariate at the natural value of the exposure H(A, W) and at the shifted value of the exposure H(A + delta, W).

est_ipcw *Estimate Inverse Probability of Censoring Weights*

Description

Estimate Inverse Probability of Censoring Weights

Usage

```
est_ipcw(V, Delta, fit_type = c("sl", "glm"), sl_learners = NULL)
```

Arguments

V	A numeric vector, matrix, data.frame or similar object giving the observed values of the covariates known to potentially inform the censoring mechanism.
Delta	A numeric vector giving observed values of the indicator function corresponding to the censoring mechanism.
fit_type	A character indicating whether to perform the fit using GLMs or a Super Learner. If use of Super Learner is desired, then the argument sl_learners must be provided.
sl_learners	An Lrnr_sl object, a Super Learner instantiated externally using **sl3**.

Details

Compute inverse probability of censoring weights for the two-phase sampling mechanism. These inverse weights are based on the probability of appearing in the second-phase sample based on variables measured on all participants.

Value

A list containing a numeric vector corresponding to the inverse probability of censoring weights required for computing an IPCW-TMLE and numeric vector of the estimated missingness mechanism. Formally, the former is nothing more than term class as computed using standard logistic regression.

est_Q *Estimate the Outcome Regression*

Description

Estimate the Outcome Regression

Usage

```
est_Q(
  Y,
  A,
  W,
  delta = 0,
  ipc_weights = rep(1, length(Y)),
  fit_type = c("sl", "glm"),
  glm_formula = "Y ~ .",
  sl_learners = NULL
)
```

Arguments

Y	A numeric vector of observed outcomes.
A	A numeric vector of observed treatment values.
W	A numeric matrix of observed baseline covariate values.
delta	A numeric indicating the magnitude of the shift to be computed for the treatment A. This is passed to the internal shift_additive and is currently limited to additive shifts.
ipc_weights	A numeric vector of observation-level weights, as produced by the internal procedure to estimate the censoring mechanism.
fit_type	A character indicating whether to use GLMs or Super Learner to fit the outcome regression. If the option "glm" is selected, the argument glm_formula must NOT be NULL, instead containing a model formula (as per glm) as a character. If the option "sl" is selected, the argument sl_learners must NOT be NULL; instead, an instantiated Lrnr_sl object, specifying learners and a metalearner for the Super Learner fit, must be provided. Consult the documentation of **sl3** for details.
glm_formula	A character corresponding to a formula to be used in fitting a generalized linear model via glm.
sl_learners	Object containing a set of instantiated learners from the **sl3**, to be used in fitting an ensemble model.

Details

Compute the outcome regression for the observed data, including with the shift imposed by the intervention. This returns the propensity score for the observed data (at A_i) and the shift (at A_i - delta).

Value

A data.table with two columns, containing estimates of the outcome mechanism at the natural value of the exposure Q(A, W) and an upshift of the exposure Q(A + delta, W).

`fit_fluctuation` *Fit One-Dimensional Fluctuation Model for Updating Initial Estimates*

Description

Fit One-Dimensional Fluctuation Model for Updating Initial Estimates

Usage

```
fit_fluctuation(
  Y,
  Qn_scaled,
  Hn,
  ipc_weights = rep(1, length(Y)),
  method = c("standard", "weighted"),
  flucmod_tol = 100
)
```

Arguments

`Y`	A `numeric` vector corresponding to an outcome variable.
`Qn_scaled`	An object providing the value of the outcome evaluate after inducing a shift in the exposure. This object should be passed in after being constructed by a call to `est_Q`.
`Hn`	An object providing values of the auxiliary ("clever") covariate, constructed from the treatment mechanism and required for targeted minimum loss estimation. This object object should be passed in after being constructed by a call to `est_Hn`.
`ipc_weights`	A `numeric` vector that gives inverse probability of censoring weights for each observation. These are generated by invoking the routines for estimating the censoring mechanism.
`method`	A `character` giving the type of regression to be used in traversing the fluctuation sub-model. The available choices are "weighted" and "standard". Consult the literature for details on the differences.
`flucmod_tol`	A `numeric` indicating the largest value to be tolerated in the fluctuation model for the targeted minimum loss estimator.

Details

Procedure for fitting a one-dimensional fluctuation model to update the initial estimates of the outcome regression based on the auxiliary covariate. These updated estimates are subsequently used to construct the TML estimator of the counterfactual mean under a modified treatment policy.

Value

A `list` containing the fluctuation model (a `glm` object) produced by logistic regression, a `character` vector indicating the type of fluctuation (whether the auxiliary covariates was used as a weight or included directly in the model formula), the updated estimates of the outcome regression under the shifted value of the exposure, and the updated estimates of the outcome regression under the natural value of exposure.

---

ipcw_eif_update *Iterative IPCW Update Procedure of Efficient Influence Function*

Description

Iterative IPCW Update Procedure of Efficient Influence Function

Usage

```
ipcw_eif_update(
  data_in,
  C,
  V,
  ipc_mech,
  ipc_weights,
  ipc_weights_norm,
  Qn_estim,
  Hn_estim,
  estimator = c("tmle", "onestep"),
  fluctuation = NULL,
  flucmod_tol = 100,
  eif_reg_type = c("hal", "glm")
)
```

Arguments

`data_in`	A `data.table` containing variables and observations of full data. That is, this corresponds to the data after application of a censoring process.
`C`	A `numeric` binary vector giving the censoring status of a given observation.
`V`	A `data.table` giving the values across all observations of all variables that play a role in the censoring mechanism.
`ipc_mech`	A `numeric` vector containing values that describe the censoring mechanism for all of the observations. Note that such values are estimated by regressing the censoring covariates `V` on the observed censoring `C` and thus correspond to predicted probabilities of being censored for each observation.
`ipc_weights`	A `numeric` vector of inverse probability of censoring weights. These are equivalent to `C / ipc_mech` in any initial run of this function. Updated values of this vector are provided as part of the output of this function, which may be used in subsequent calls that allow convergence to a more efficient estimate.

`ipc_weights_norm`
: A `numeric` vector of the weights described in the previous argument. In this case, the weights are normalized.

`Qn_estim`
: A `data.table` corresponding to the outcome regression. This is produced by invoking the internal function `est_Q`.

`Hn_estim`
: A `data.table` corresponding to values produced in the computation of the auxiliary ("clever") covariate. This is produced easily by invoking the internal function `est_Hn`.

`estimator`
: The type of estimator to be fit, either `"tmle"` for targeted maximum likelihood estimation or `"onestep"` for a one-step estimator.

`fluctuation`
: A `character` giving the type of regression to be used in traversing the fluctuation submodel. The choices are "weighted" and "standard".

`flucmod_tol`
: A `numeric` indicating the largest value to be tolerated in the fluctuation model for the targeted minimum loss estimator.

`eif_reg_type`
: Whether a flexible nonparametric function ought to be used in the dimension-reduced nuisance regression of the targeting step for the censored data case. By default, the method used is a nonparametric regression based on the Highly Adaptive Lasso (from **hal9001**). Set this to `"glm"` to instead use a simple linear regression model. In this step, the efficient influence function (EIF) is regressed against covariates contributing to the censoring mechanism (i.e., EIF ~ V | C = 1).

Details

An adaptation of the IPCW-TMLE for iteratively constructing an efficient inverse probability of censoring weighted TML or one-step estimator. The efficient influence function of the parameter and updating the IPC weights in an iterative process, until a convergence criteria is satisfied.

Value

A `list` containing the estimated outcome mechanism, the fitted fluctuation model for TML updates, the updated inverse probability of censoring weights (IPCW), normalized versions of the same weights, the updated estimate of the efficient influence function, and the estimated IPCW component of the EIF.

`msm_vimshift` *Working marginal structural model for causal effects of an intervention grid*

Description

Working marginal structural model for causal effects of an intervention grid

Usage

```
msm_vimshift(
  Y,
  A,
  W,
  C = rep(1, length(Y)),
  V = NULL,
  delta_grid = seq(-0.5, 0.5, 0.5),
  msm_form = list(type = "linear", knot = NA),
  estimator = c("tmle", "onestep"),
  weighting = c("identity", "variance"),
  ci_level = 0.95,
  ci_type = c("marginal", "simultaneous"),
  ...
)
```

Arguments

Y	A numeric vector of the observed outcomes.
A	A numeric vector corresponding to a treatment variable. The parameter of interest is defined as a location shift of this quantity.
W	A matrix, data.frame, or similar corresponding to a set of baseline covariates.
C	A numeric indicator for whether a given observation was subject to censoring in the two-phase sample. This is used to compute an IPCW-TMLE in such cases. The default assumes no censoring.
V	The covariates that are used in determining the sampling procedure that gives rise to censoring. The default is NULL and corresponds to scenarios in which there is no censoring (in which case all values in the preceding argument C must be uniquely 1. To specify this, pass in a NAMED list identifying variables amongst W, A, Y that are thought to have played a role in defining the sampling/censoring mechanism (C).
delta_grid	A numeric vector giving the individual values of the shift parameter used in computing each of the estimates.
msm_form	A list specifying the type of working MSM to fit to summarize the counterfactual means. The list has two components: (1) "type", which may be either "linear" or "piecewise", and (2) "knot", which, if specified, must be a value in delta_grid. See examples for its use.
estimator	The type of estimator to be fit, either "tmle" for targeted maximum likelihood estimation or "onestep" for a one-step augmented inverse probability weighted (AIPW) estimator.
weighting	Whether to weight each parameter estimate by the inverse of its variance (in order to improve stability of the resultant MSM fit) or to simply weight all parameter estimates equally. The default is the option "identity", weighting all estimates identically.
ci_level	A numeric indicating the desired coverage level of the confidence interval to be computed.

ci_type	Whether to construct a simultaneous confidence band covering all parameter estimates at once or marginal confidence intervals covering each parameter estimate separately. The default is to construct marginal confidence intervals for each parameter estimate rather than a simultaneous confidence band.
...	Additional arguments to be passed to txshift.

Details

Computes estimates of the counterfactual mean over a grid of shift stochastic interventions and fits a working marginal structural model to summarize the trend through the counterfactual means as a function of the specified shift intervention. The working marginal structural model may be linear in the shift parameter or piecewise linear with a single knot point. Provides support for two weighting schemes, may be used with either of the one-step or TML estimators, and also allows the construction of marginal or simultaneous confidence intervals.

Value

A list containing estimates of the individual counterfactual means over a grid in the shift parameters (delta_grid), alongside the estimate of a marginal structural model that summarizes a trend through these counterfactual means.

Examples

```
if (require("sl3")) {
  n_obs <- 100
  W <- as.numeric(replicate(1, rbinom(n_obs, 1, 0.5)))
  A <- as.numeric(rnorm(n_obs, mean = 2 * W, sd = 1))
  Y <- rbinom(n_obs, 1, plogis(2 * A - W))
  msm <- msm_vimshift(
    W = W, A = A, Y = Y, estimator = "tmle",
    g_fit_args = list(
      fit_type = "sl",
      sl_learners_density = Lrnr_density_hse$new(Lrnr_glm$new())
    ),
    Q_fit_args = list(
      fit_type = "glm",
      glm_formula = "Y ~ ."
    ),
    delta_grid = seq(-1, 1, 0.25)
  )

  # fit a linear spline with knot at 0
  n_obs <- 100
  W <- as.numeric(replicate(1, rbinom(n_obs, 1, 0.5)))
  A <- as.numeric(rnorm(n_obs, mean = 2 * W, sd = 1))
  Y <- rbinom(n_obs, 1, plogis(0.1 * A * (A >= 0) - 3 * A * (A < 0) - W))
  msm <- msm_vimshift(
    W = W, A = A, Y = Y, estimator = "tmle",
    g_fit_args = list(
      fit_type = "sl",
      sl_learners_density = Lrnr_density_hse$new(Lrnr_glm$new())
```

```
    ),
    Q_fit_args = list(
      fit_type = "glm",
      glm_formula = "Y ~ ."
    ),
    delta_grid = seq(-1, 1, 0.25),
    msm_form = list(type = "piecewise", knot = 0)
  )
}
```

`onestep_txshift` *Compute One-Step Estimate of Counterfactual Mean Under Stochastic Shift Intervention*

Description

Compute One-Step Estimate of Counterfactual Mean Under Stochastic Shift Intervention

Usage

```
onestep_txshift(
  data_internal,
  C = rep(1, nrow(data_internal)),
  V = NULL,
  delta,
  ipcw_estim,
  Qn_estim,
  Hn_estim,
  eif_reg_type = c("hal", "glm"),
  ipcw_fit_args,
  ipcw_efficiency = TRUE
)
```

Arguments

`data_internal`	A `data.table` constructed internally by a call to `txshift`. This contains the data elements needed for computing the one-step estimator.
`C`	A `numeric` indicator for whether a given observation censored in the two-phase sampling procedure, used to compute an IPC-weighted one-step estimator in cases where two-stage sampling is performed. Default assumes no censoring.
`V`	The covariates that are used in determining the sampling procedure that gives rise to censoring. The default is `NULL` and corresponds to scenarios in which there is no censoring (in which case all values in the preceding argument `C` must be uniquely 1. To specify this, pass in a NAMED `list` identifying variables amongst W, A, Y that are thought to have played a role in defining the sampling/censoring mechanism (C).

`delta` A `numeric` value indicating the shift in the treatment to be used in defining the target parameter. This is defined with respect to the scale of the treatment (A).

`ipcw_estim` An object providing the value of the censoring mechanism evaluated across the full data. This object is passed in after being constructed by a call to the internal function `est_ipcw`.

`Qn_estim` An object providing the value of the outcome evaluated after imposing a shift in the treatment. This object is passed in after being constructed by a call to the internal function `est_Q`.

`Hn_estim` An object providing values of the auxiliary ("clever") covariate, constructed from the treatment mechanism and required for targeted minimum loss estimation. This object object should be passed in after being constructed by a call to the internal function `est_Hn`.

`eif_reg_type` Whether a flexible nonparametric function ought to be used in the dimension-reduced nuisance regression of the targeting step for the censored data case. By default, the method used is a nonparametric regression based on the Highly Adaptive Lasso (from **hal9001**). Set this to `"glm"` to instead use a simple linear regression model. In this step, the efficient influence function (EIF) is regressed against covariates contributing to the censoring mechanism (i.e., EIF ~ V | C = 1).

`ipcw_fit_args` A `list` of arguments, all but one of which are passed to `est_ipcw`. For details, consult the documentation for `est_ipcw`. The first element (i.e., `fit_type`) is used to determine how this regression is fit: "glm" for generalized linear model, "sl" for a Super Learner, and "external" for a user-specified input of the form produced by `est_ipcw`.

`ipcw_efficiency` Whether to invoke an augmentation of the IPCW-TMLE procedure that performs an iterative process to ensure efficiency of the resulting estimate. The default is `TRUE`; set to `FALSE` to use an IPC-weighted loss rather than the IPC-augmented influence function.

Details

Invokes the procedure to construct a one-step estimate of the counterfactual mean under a modified treatment policy.

Value

S3 object of class `txshift` containing the results of the procedure to compute a one-step estimate of the treatment shift parameter.

`plot.txshift_msm` *Plot working MSM for causal effects of an intervention grid*

Description

Plot working MSM for causal effects of an intervention grid

Usage

```
## S3 method for class 'txshift_msm'
plot(x, ...)
```

Arguments

x	Object of class `txshift_msm` as produced by a call to `msm_vimshift`.
...	Additional arguments passed to `plot` as necessary.

Details

Creates a visualization of the intervention-specific counterfactual means as well as the working marginal structural model summarizing the trend across posited values of the intervention.

Examples

```
if (require("sl3")) {
  set.seed(3287)
  n_obs <- 1000
  W <- as.numeric(replicate(1, rbinom(n_obs, 1, 0.5)))
  A <- as.numeric(rnorm(n_obs, mean = 2 * W, sd = 1))
  Y <- rbinom(n_obs, 1, plogis(2 * A - W))
  msm <- msm_vimshift(
    W = W, A = A, Y = Y, estimator = "tmle",
    g_fit_args = list(
      fit_type = "sl",
      sl_learners_density = Lrnr_density_hse$new(Lrnr_glm$new())
    ),
    Q_fit_args = list(
      fit_type = "glm",
      glm_formula = "Y ~ ."
    ),
    delta_grid = seq(-1, 1, 0.25)
  )
  plot(msm)

  # fit a linear spline with knot at 0
  set.seed(8293)
  n_obs <- 1000
  W <- as.numeric(replicate(1, rbinom(n_obs, 1, 0.5)))
  A <- as.numeric(rnorm(n_obs, mean = 2 * W, sd = 1))
  Y <- rbinom(n_obs, 1, plogis(0.1 * A * (A >= 0) - 3 * A * (A < 0) - W))
  msm <- msm_vimshift(
    W = W, A = A, Y = Y, estimator = "tmle",
    g_fit_args = list(
      fit_type = "sl",
      sl_learners_density = Lrnr_density_hse$new(Lrnr_glm$new())
    ),
    Q_fit_args = list(
      fit_type = "glm",
      glm_formula = "Y ~ ."
```

```
    ),
    delta_grid = seq(-1, 1, 0.25),
    msm_form = list(type = "piecewise", knot = 0)
  )
  plot(msm)
}
```

`print.txshift` *Print Method for Counterfactual Means*

Description

Print Method for Counterfactual Means

Usage

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

Arguments

`x`	An object of class `txshift`.
`...`	Other options (not currently used).

Details

The `print` method for objects of class `txshift`.

Value

None. Called for the side effect of printing particular slots of objects of class `txshift`.

Examples

```
set.seed(429153)
n_obs <- 100
W <- replicate(2, rbinom(n_obs, 1, 0.5))
A <- rnorm(n_obs, mean = 2 * W, sd = 1)
Y <- rbinom(n_obs, 1, plogis(A + W + rnorm(n_obs, mean = 0, sd = 1)))
txout <- txshift(
  W = W, A = A, Y = Y, delta = 0.5,
  estimator = "tmle",
  g_fit_args = list(
    fit_type = "hal", n_bins = 5,
    grid_type = "equal_mass",
    lambda_seq = exp(-1:-9)
  ),
  Q_fit_args = list(
    fit_type = "glm",
```

```
    glm_formula = "Y ~ ."
  )
)
print(txout)
```

`print.txshift_msm` *Print Method for Marginal Structural Models*

Description

Print Method for Marginal Structural Models

Usage

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

Arguments

`x`	An object of class `txshift_msm`.
`...`	Other options (not currently used).

Details

The `print` method for objects of class `txshift_msm`.

Value

None. Called for the side effect of printing particular slots of objects of class `txshift_msm`.

Examples

```
if (require("sl3")) {
  set.seed(3287)
  n_obs <- 1000
  W <- as.numeric(replicate(1, rbinom(n_obs, 1, 0.5)))
  A <- as.numeric(rnorm(n_obs, mean = 2 * W, sd = 1))
  Y <- rbinom(n_obs, 1, plogis(2 * A - W))
  msm <- msm_vimshift(
    W = W, A = A, Y = Y, estimator = "tmle",
    g_fit_args = list(
      fit_type = "sl",
      sl_learners_density = Lrnr_density_hse$new(Lrnr_glm$new())
    ),
    Q_fit_args = list(
      fit_type = "glm",
      glm_formula = "Y ~ ."
    ),
    delta_grid = seq(-1, 1, 0.25)
```

```
  )
  print(msm)
}
```

`scale_to_original` *Transform values from the unit interval back to their original scale*

Description

Transform values from the unit interval back to their original scale

Usage

```
scale_to_original(scaled_vals, max_orig, min_orig)
```

Arguments

`scaled_vals`	A `numeric` vector corresponding to re-scaled values in the unit interval, to be re-scaled to the original interval.
`max_orig`	A `numeric` scalar value giving the maximum of the values on the original scale.
`min_orig`	A `numeric` scalar value giving the minimum of the values on the original scale.

Details

A back-transformation that returns values computed in the unit interval to their original scale. This is used in re-scaling updated TML estimates back to their natural scale. Undoes `scale_to_unit`.

Value

A `numeric` vector of the same length as `scaled_vals`, where the values are re-scaled to lie in their original/natural interval.

`scale_to_unit` *Transform values by scaling to the unit interval*

Description

Transform values by scaling to the unit interval

Usage

```
scale_to_unit(vals)
```

Arguments

`vals`	A `numeric` vector corresponding to the observed values of the variable of interest, to be re-scaled to the unit interval [0,1].

Details

A transformation that scales an arbitrary set of input values to the unit interval. See scale_to_original for a corresponding backtransformation.

Value

A numeric vector of the same length as vals, where the values are re-scaled to lie in unit interval [0, 1].

---

shift_additive *Simple Additive Modified Treatment Policy*

Description

Simple Additive Modified Treatment Policy

Usage

```
shift_additive(A, W = NULL, delta)
```

Arguments

A	A numeric vector of observed treatment values.
W	A numeric matrix of observed baseline covariate values.
delta	A numeric indicating the magnitude of the shift to be computed for the treatment A.

Details

A simple modified treatment policy that modifes the observed value of the exposure by shifting it by a value delta. Note that this shifting function assumes support of A|W across all strata of W.

Value

A numeric vector containing the shifted exposure values.

summary.txshift *Summary for Counterfactual Mean Under Stochastic Intervention*

Description

Summary for Counterfactual Mean Under Stochastic Intervention

Usage

```
## S3 method for class 'txshift'
summary(object, ..., ci_level = 0.95, digits = 4)
```

Arguments

object	An object of class txshift, as produced by invoking the function txshift, for which a confidence interval is to be computed.
...	Other arguments. Not currently used.
ci_level	A numeric indicating the level of the confidence interval to be computed.
digits	A numeric scalar giving the number of digits to be displayed or to round results to.

Details

Print a convenient summary for objects computed using txshift.

Value

None. Called for the side effect of printing a summary of particular slots of objects of class txshift.

Examples

```
set.seed(429153)
n_obs <- 100
W <- replicate(2, rbinom(n_obs, 1, 0.5))
A <- rnorm(n_obs, mean = 2 * W, sd = 1)
Y <- rbinom(n_obs, 1, plogis(A + W + rnorm(n_obs, mean = 0, sd = 1)))
txout <- txshift(
  W = W, A = A, Y = Y, delta = 0.5,
  estimator = "tmle",
  g_fit_args = list(
    fit_type = "hal", n_bins = 5,
    grid_type = "equal_mass",
    lambda_seq = exp(-1:-9)
  ),
  Q_fit_args = list(
    fit_type = "glm",
    glm_formula = "Y ~ ."
  )
```

```
)
summary(txout)
```

`tmle_txshift` *Compute Targeted Minimum Loss Estimate of Counterfactual Mean Under Stochastic Shift Intervention*

Description

Compute Targeted Minimum Loss Estimate of Counterfactual Mean Under Stochastic Shift Intervention

Usage

```
tmle_txshift(
  data_internal,
  C = rep(1, nrow(data_internal)),
  V = NULL,
  delta,
  ipcw_estim,
  Qn_estim,
  Hn_estim,
  fluctuation = c("standard", "weighted"),
  max_iter = 10,
  eif_reg_type = c("hal", "glm"),
  ipcw_fit_args,
  ipcw_efficiency = TRUE
)
```

Arguments

`data_internal`	A `data.table` constructed internally by a call to `txshift`. This contains most of the data for computing the TML estimator.
`C`	A `numeric` indicator for whether a given observation was subject to censoring, used to compute an IPCW-TMLE in cases where two-stage sampling is performed. Default assumes no censoring.
`V`	The covariates that are used in determining the sampling procedure that gives rise to censoring. The default is `NULL` and corresponds to scenarios in which there is no censoring (in which case all values in the preceding argument `C` must be uniquely 1. To specify this, pass in a NAMED `list` identifying variables amongst W, A, Y that are thought to have played a role in defining the sampling/censoring mechanism (C).
`delta`	A `numeric` value indicating the shift in the treatment to be used in defining the target parameter. This is defined with respect to the scale of the treatment (A).
`ipcw_estim`	An object providing the value of the censoring mechanism evaluated across the full data. This object is passed in after being constructed by a call to the internal function `est_ipcw`.

`Qn_estim` An object providing the value of the outcome evaluated after imposing a shift in the treatment. This object is passed in after being constructed by a call to the internal function `est_Q`.

`Hn_estim` An object providing values of the auxiliary ("clever") covariate, constructed from the treatment mechanism and required for targeted minimum loss-based estimation. This object object should be passed in after being constructed by a call to `est_Hn`.

`fluctuation` The method to be used in the submodel fluctuation step (targeting step) to compute the TML estimator. The choices are "standard" and "weighted" for where to place the auxiliary covariate in the logistic tilting regression.

`max_iter` A `numeric` integer giving the maximum number of steps to be taken in iterating to a solution of the efficient influence function.

`eif_reg_type` Whether a flexible nonparametric function ought to be used in the dimension-reduced nuisance regression of the targeting step for the censored data case. By default, the method used is a nonparametric regression based on the Highly Adaptive Lasso (from **hal9001**). Set this to `"glm"` to instead use a simple linear regression model. In this step, the efficient influence function (EIF) is regressed against covariates contributing to the censoring mechanism (i.e., EIF ~ V | C = 1).

`ipcw_fit_args` A `list` of arguments, all but one of which are passed to `est_ipcw`. For details, consult the documentation for `est_ipcw`. The first element (i.e., `fit_type`) is used to determine how this regression is fit: "glm" for generalized linear model, "sl" for a Super Learner, and "external" for a user-specified input of the form produced by `est_ipcw`.

`ipcw_efficiency` Whether to invoke an augmentation of the IPCW-TMLE procedure that performs an iterative process to ensure efficiency of the resulting estimate. The default is `TRUE`; set to `FALSE` to use an IPC-weighted loss rather than the IPC-augmented influence function.

Details

Invokes the procedure to construct a targeted minimum loss estimate (TMLE) of the counterfactual mean under a modified treatment policy.

Value

S3 object of class `txshift` containing the results of the procedure to compute a TML estimate of the treatment shift parameter.

`txshift` *Estimate Counterfactual Mean Under Stochastic Shift in Exposure*

Description

Estimate Counterfactual Mean Under Stochastic Shift in Exposure

Usage

```
txshift(
  W,
  A,
  Y,
  C = rep(1, length(Y)),
  V = NULL,
  delta = 0,
  estimator = c("tmle", "onestep"),
  fluctuation = c("standard", "weighted"),
  max_iter = 10,
  ipcw_fit_args = list(fit_type = c("glm", "sl", "external"), sl_learners = NULL),
  g_fit_args = list(fit_type = c("hal", "sl", "external"), n_bins = c(10, 25),
   grid_type = c("equal_range", "equal_range"), lambda_seq = exp(seq(-1, -13, length =
    300)), use_future = FALSE, sl_learners_density = NULL),
  Q_fit_args = list(fit_type = c("glm", "sl", "external"), glm_formula = "Y ~ .",
    sl_learners = NULL),
  eif_reg_type = c("hal", "glm"),
  ipcw_efficiency = TRUE,
  ipcw_fit_ext = NULL,
  gn_fit_ext = NULL,
  Qn_fit_ext = NULL
)
```

Arguments

W	A matrix, data.frame, or similar containing a set of baseline covariates.
A	A numeric vector corresponding to a treatment variable. The parameter of interest is defined as a location shift of this quantity.
Y	A numeric vector of the observed outcomes.
C	A numeric indicator for whether a given observation was subject to censoring, used to compute an IPC-weighted estimator in cases where two-stage sampling is performed. The default assumes no censoring.
V	The covariates that are used in determining the sampling procedure that gives rise to censoring. The default is NULL and corresponds to scenarios in which there is no censoring (in which case all values in the preceding argument C must be uniquely 1). To specify this, pass in a character vector identifying variables amongst W, A, Y thought to have played a role in defining the sampling/censoring mechanism (C). This argument also accepts a data.table (or similar) object composed of combinations of variables W, A, Y; use of this option is NOT recommended.
delta	A numeric value indicating the shift in the treatment to be used in defining the target parameter. This is defined with respect to the scale of the treatment (A).
estimator	The type of estimator to be fit, either "tmle" for targeted maximum likelihood or "onestep" for a one-step estimator.

`fluctuation` The method to be used in the submodel fluctuation step (targeting step) to compute the TML estimator. The choices are "standard" and "weighted" for where to place the auxiliary covariate in the logistic tilting regression.

`max_iter` A `numeric` integer giving the maximum number of steps to be taken in iterating to a solution of the efficient influence function.

`ipcw_fit_args` A `list` of arguments, all but one of which are passed to `est_ipcw`. For details, consult the documentation of `est_ipcw`. The first element (i.e., `fit_type`) is used to determine how this regression is fit: generalized linear model ("glm") or Super Learner ("sl"), and "external" a user-specified input of the form produced by `est_ipcw`. NOTE THAT this first argument is not passed to `est_ipcw`.

`g_fit_args` A `list` of arguments, all but one of which are passed to `est_g`. For details, consult the documentation of `est_g`. The first element (i.e., `fit_type`) is used to determine how this regression is fit: `"hal"` to estimate conditional densities via the highly adaptive lasso (via **haldensify**), `"sl"` for **sl3** learners used to fit Super Learner to densities via `Lrnr_haldensify` or similar, and `"external"` for user-specified input of the form produced by `est_g`. NOTE that this first argument is not passed to `est_g`.

`Q_fit_args` A `list` of arguments, all but one of which are passed to `est_Q`. For details, consult the documentation for `est_Q`. The first element (i.e., `fit_type`) is used to determine how this regression is fit: `"glm"` for a generalized linear model for the outcome regression, `"sl"` for **sl3** learners used to fit a Super Learner for the outcome regression, and `"external"` for user-specified input of the form produced by `est_Q`. NOTE that this first argument is not passed to `est_g`.

`eif_reg_type` Whether a flexible nonparametric function ought to be used in the dimension-reduced nuisance regression of the targeting step for the censored data case. By default, the method used is a nonparametric regression based on the Highly Adaptive Lasso (from **hal9001**). Set this to `"glm"` to instead use a simple linear regression model. In this step, the efficient influence function (EIF) is regressed against covariates contributing to the censoring mechanism (i.e., EIF ~ V | C = 1).

`ipcw_efficiency` Whether to invoke an augmentation of the IPCW-TMLE procedure that performs an iterative process to ensure efficiency of the resulting estimate. The default is `TRUE`; only set to `FALSE` if possible inefficiency of the IPCW-TMLE is not a concern.

`ipcw_fit_ext` The results of an external fitting procedure used to estimate the two-phase censoring mechanism, to be used in constructing the inverse probability of censoring weighted TML or one-step estimator. The input provided must match the output of `est_ipcw` exactly; thus, use of this argument is only recommended for power users.

`gn_fit_ext` The results of an external fitting procedure used to estimate the exposure mechanism (generalized propensity score), to be used in constructing the TML or one-step estimator. The input provided must match the output of `est_g` exactly; thus, use of this argument is only recommended for power users.

`Qn_fit_ext` The results of an external fitting procedure used to estimate the outcome mechanism, to be used in constructing the TML or one-step estimator. The input

provided must match the output of est_Q exactly; thus, use of this argument is only recommended for power users.

Details

Construct a one-step estimate or targeted minimum loss estimate of the counterfactual mean under a modified treatment policy, automatically making adjustments for two-phase sampling when a censoring indicator is included. Ensemble machine learning may be used to construct the initial estimates of nuisance functions using **sl3**.

Value

S3 object of class `txshift` containing the results of the procedure to compute a TML or one-step estimate of the counterfactual mean under a modified treatment policy that shifts a continuous-valued exposure by a scalar amount `delta`. These estimates can be augmented to be consistent and efficient when two-phase sampling is performed.

Examples

```
set.seed(429153)
n_obs <- 100
W <- replicate(2, rbinom(n_obs, 1, 0.5))
A <- rnorm(n_obs, mean = 2 * W, sd = 1)
Y <- rbinom(n_obs, 1, plogis(A + W + rnorm(n_obs, mean = 0, sd = 1)))
C <- rbinom(n_obs, 1, plogis(W + Y)) # two-phase sampling

# construct a TML estimate (set estimator = "onestep" for the one-step)
tmle <- txshift(
  W = W, A = A, Y = Y, delta = 0.5,
  estimator = "tmle",
  g_fit_args = list(
    fit_type = "hal", n_bins = 5,
    grid_type = "equal_range",
    lambda_seq = exp(-1:-9)
  ),
  Q_fit_args = list(
    fit_type = "glm",
    glm_formula = "Y ~ ."
  )
)

# construct a TML estimate under two-phase sampling
ipcwtmle <- txshift(
  W = W, A = A, Y = Y, delta = 0.5,
  C = C, V = c("W", "Y"),
  estimator = "tmle", max_iter = 5,
  ipcw_fit_args = list(fit_type = "glm"),
  g_fit_args = list(
    fit_type = "hal", n_bins = 5,
    grid_type = "equal_range",
    lambda_seq = exp(-1:-9)
  ),
```

```
  Q_fit_args = list(
    fit_type = "glm",
    glm_formula = "Y ~ ."
  ),
  eif_reg_type = "glm"
)
```

Index