Package 'DTRreg'

May 7, 2026

Type Package

Title DTR Estimation and Inference via G-Estimation, Dynamic WOLS, Q-Learning, and Dynamic Weighted Survival Modeling (DWSurv)

Version 2.3

Date 2025-05-03

Author Michael Wallace [aut],
Erica E M Moodie [aut],
David A Stephens [aut],
Gabrielle Simoneau [aut],
Shannon T. Holloway [aut, cre],
Juliana Schulz [aut]

Maintainer Shannon T. Holloway <shannon.t.holloway@gmail.com>

Description Dynamic treatment regime estimation and inference via G-estimation, dynamic weighted ordinary least squares (dWOLS) and Q-learning. Inference via bootstrap and recursive sandwich estimation. Estimation and inference for survival outcomes via Dynamic Weighted Survival Modeling (DWSurv). Extension to continuous treatment variables. Wallace et al. (2017) <DOI:10.18637/jss.v080.i02>; Simoneau et al. (2020) <DOI:10.1080/00949655.2020.1793341>.

License GPL-2

Depends R (>= 4.1.0)

Imports graphics, nnet, R6, stats, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Repository CRAN

Collate 'treatmentClasses.R' 'Ahat.R' 'interactive.R' 'inputProcessing.R' 'utils.R' 'weights.R' 'sandwich.R' 'gest.R' 'dwols.R' 'bootstrapStep.R' 'dtrProcedure.R' 'DTRreg.R' 'DWsurv.R' 'chooseM.R' 'data_generate.R' 's3-methods.R' 'varest.R'

Date/Publication 2025-05-03 18:50:02 UTC

Contents

`chooseM` *Adaptive Choice of the Bootstrap Resample Size M*

Description

Implementation of a double-bootstrap algorithm for choosing the bootstrap resample size m in a data-adaptive manner. The function returns the resample size to be used to apply the m-out-of-n bootstrap with DTRreg.

Usage

```
chooseM(
  outcome,
  blip.mod,
  treat.mod,
  tf.mod,
  data = NULL,
  method = c("gest", "dwols", "qlearn"),
  treat.type = c("bin", "multi", "cont"),
  treat.fam = gaussian(link = "identity"),
  weight = c("abs", "ipw", "cipw", "qpom", "wo", "none", "manual"),
  n.bins = 3L,
  treat.wgt.man = NULL,
  treat.range = NULL,
  missing = c("drop", "ipw"),
  missing.mod = NULL,
  B1 = 500,
  B2 = 500
)
```

Arguments

`outcome`	The outcome variable. Missing data will result in a stopping error.
`blip.mod`	A list of formula objects specifying covariates of the blip function for each stage in order. No dependent variable should be specified. Note that this input should include the treatment variable ONLY if the blip model is quadratic in treatment. See Details for further clarification.

`treat.mod` A list of formula objects specifying the treatment model for each stage in order. Treatment variable should be included as the dependent variable. If treatment is binary `glm(family = binomial)` will be used to obtain parameter estimates; if treatment is multi-nomial, `multinom()` will be used to obtain parameter estimates; and if treatment is continuous, `lm()` will be used.

`tf.mod` A list of formula objects specifying covariates of the treatment-free model for each stage in order. No dependent variable should be specified.

`data` A data frame containing all necessary covariates and treatments contained in the models. Missing data should be coded as `NA`.

`method` The DTR method to be used, choose "dwols" for dynamic WOLS, "gest" for G-estimation, or "qlearn" for Q-learning.

`treat.type` A character object. Must be one of {"bin", "multi", "cont"} indicating that the treatments at each stage are binary, multinomial, or continuous, respectively. Each stage must have the same treatment type.

`treat.fam` A character or family object. The description of the dose distribution along with the link function to be used in the treatment model for computing weights; should be specified in a similar format as that used in `glm()`. If character object, must be one of {"gaussian", "Gamma"}, for which `gaussian(link = "identity")` or `Gamma(link = "log")` will be used, respectively. Input is ignored for `treat.type = "bin"` and `treat.type = "multi"`.

`weight` The form of the treatment weight. See details.

`n.bins` An integer object. The number of bins (levels) to be used for categorizing continuous doses. This input is required only when `treat.type = "cont"` and `weight = "wo"` or `weight = "qpom"`.

`treat.wgt.man` NULL or a list of vectors of known treatment weights can be specified to be used instead of hard-coded treatment weight options. The i^{th} element of the list contains the multiplicative weights for the i^{th} stage. Each vector must be of length n, the number of participants. Used only for `method = "dwols"`.

`treat.range` For continuous treatments. Specify the maximum/minimum value that treatments can be take. If unspecified then the minimum/maximum value of observed treatments is used. If you wish to have unrestricted treatments set this option to `c(-Inf, +Inf)`. If each stage has its own range, provide as a list, the ith element providing the min and max for the ith stage treatment.

`missing` A character object. Must be one of {"drop", "ipw"}. If set to "ipw" and covariate or treatment data are missing then inverse probability weights are used. The complete case probability is estimated via logistic regression. If set to "drop" and data are missing, participants with missing data are dropped for all stage analyses.

`missing.mod` An optional list of formula objects specifying the model for the inverse probability of weights for each stage in order. No dependent variable should be specified. If `missing = "ipw"` and `missing.mod = NULL`, then the models are assumed to be linear comprising the full covariate history derived from all of the previous stage models.

`B1` Number of first-level bootstrap resamples.

`B2` Number of second-level bootstrap resamples.

Details

The m-out-of-n bootstrap is an adequate tool for constructing valid confidence intervals for the first stage parameters in DTRreg. The resample size m is: $m = n^{\frac{1+\alpha(1-\hat{p})}{1+\alpha}}$. The estimated non-regularity level is computed by DTRreg. The double-bootstrap algorithm is a cross-validation tool for choosing the tuning parameter α in a data-driven way.

The current implementation is valid for a two-stage DTR. Moreover, the current implementation may be unstable when there are many missing data.

Value

A list with a single element

`m` Resample size for using in the m-out-of-n bootstrap.

Author(s)

Gabrielle Simoneau

References

Chakraborty, B., Moodie, E. E. M. (2013) *Statistical Methods for Dynamic Treatment Regimes*. New York: Springer.

Efron B., Tibshirani R. J. (1994) An Introduction to the Bootstrap. *CRC press*.

Wallace, M. P., Moodie, E. M. (2015) Doubly-Robust Dynamic Treatment Regimen Estimation Via Weighted Least Squares. *Biometrics* **71**(3), 636–644 (doi:10.1111/biom.12306.)

Examples

```
data(twoStageCont)

# models to be passed to DTRreg
# blip model
blip.mod <- list(~ X1, ~ X2)
# treatment model (correctly specified)
treat.mod <- list(A1 ~ X1, A2 ~ 1)
# treatment-free model (incorrectly specified)
tf.mod <- list(~ X1, ~ X2)

# perform dWOLS without calculating confidence intervals
mod1 <- DTRreg(twoStageCont$Y, blip.mod, treat.mod, tf.mod,
               data = twoStageCont, method = "dwols")

# choose m adaptively for that model
## Not run:
  m <- chooseM(twoStageCont$Y, blip.mod, treat.mod, tf.mod,
               data = twoStageCont, method = "dwols",
               B1 = 200, B2 = 200)$m

## End(Not run)
m <- 94
```

```
# dWOLS with confidence intervals from the m-out-of-n bootstrap
mod2 <- DTRreg(twoStageCont$Y, blip.mod, treat.mod, tf.mod,
               data = twoStageCont, method = "dwols",
               var.estim = "bootstrap",
               bootstrap.controls = list(M = m))
```

`confint.DTRreg` *Confidence Interval Calculations for DTRs*

Description

Confidence intervals for parameters, with the option of constructing the confidence intervals using the percentile method when bootstrap is used.

Usage

```
## S3 method for class 'DTRreg'
confint(object, parm = NULL, level = 0.95, type = c("se", "percentile"), ...)
```

Arguments

`object`	A model object generated by the function DTRreg.
`parm`	Not available for DTRreg objects.
`level`	The confidence level required.
`type`	Typical Wald-type confidence interval "se" (default) or confidence intervals derived with the percentile method "percentile" (bootstrap variance estimates only).
`...`	Space for additional arguments (not currently used).

Value

A list with columns giving lower and upper confidence limits for each parameter. These will be labelled as (1-level)/2 and 1 - (1-level)/2 in percentage (by default 2.5% and 97.5%).

`data` *Toy Two-Stage Trial Datasets*

Description

These datasets are provided only to facilitate examples. They are not based on or representative of any real-world applications.

Usage

```
data(twoStageCont)

data(twoStageCens)

data(twoStageSurv)
```

Format

twoStageCont is a dataset generated to mimic a simple two-stage trial. The data.frame contains 1000 observations with 5 columns:

X1 The first stage covariate. A normally distributed continuous variable.

A1 The first stage treatment. A binary variable.

X2 The second stage covariate. A normally distributed continuous variable.

A2 The second stage treatment. A binary variable.

Y The outcome. A continuous variable.

twoStageCens is a dataset generated to mimic a simple two-stage trial with right-censoring. The data.frame contains 1000 observations with 9 columns:

X11 A first stage covariate. A normally distributed continuous variable.

X12 A first stage covariate. A continuous variable = X11^4.

A1 The first stage treatment. A binary variable.

T1 The time from the beginning of the first stage to the event or to stage 2 entry, whichever comes first.

X21 A second stage covariate. A normally distributed continuous variable.

X22 A second stage covariate. A continuous variable = X21^3.

A2 The second stage treatment. A binary variable.

T2 The time from the beginning of the second stage to the event defined only for subjects who enter the second stage.

delta Event indicator.

Note: For participants who experienced the event during stage 1, i.e., did not continue to stage 2, the "survival time" is T1. For participants that entered stage 2, the "survival time" is T1 + T2.

twoStageSurv is a dataset generated to mimic a simple two-stage trial without censoring. The data.frame contains 1000 observations with 9 columns:

X11 A first stage covariate. A normally distributed continuous variable.

X12 A first stage covariate. A continuous variable = X11^4.

A1 The first stage treatment. A binary variable.

T1 The time from the beginning of the first stage to the event or to stage 2 entry, whichever comes first.

X21 A second stage covariate. A normally distributed continuous variable.

X22 A second stage covariate. A continuous variable = X21^3.

A2 The second stage treatment. A binary variable.

T2 The time from the beginning of the second stage to the event non-zero only for subjects who did not have an event in Stage I.

Note: The "survival time" is T1 + T2.

`DTRreg` *DTR Estimation and Inference via G-estimation, Dynamic WOLS, or Q-learning*

Description

Dynamic treatment regimen estimation and inference via G-estimation and dynamic WOLS. Estimation of blip model parameters for multi-stage data.

Usage

```
DTRreg(
  outcome,
  blip.mod,
  treat.mod,
  tf.mod,
  data = NULL,
  method = c("gest", "dwols", "qlearn"),
  interactive = FALSE,
  treat.type = c("bin", "multi", "cont"),
  treat.fam = gaussian(link = "identity"),
  weight = c("abs", "ipw", "cipw", "qpom", "wo", "none", "manual"),
  n.bins = 3L,
  treat.range = NULL,
  treat.wgt.man = NULL,
  var.estim = c("none", "bootstrap", "sandwich"),
  full.cov = FALSE,
 bootstrap.controls = list(B = 100L, M = nrow(data), type = "standard", truncate = 0,
    verbose = FALSE, interrupt = FALSE),
  missing = c("drop", "ipw"),
  missing.mod = NULL,
  dtr = TRUE
)

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

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

```
## S3 method for class 'DTRreg'
coef(object, ...)
```

Arguments

outcome	The outcome variable. Missing data will result in a stopping error.
blip.mod	A list of formula objects specifying covariates of the blip function for each stage in order. No dependent variable should be specified. Note that this input should include the treatment variable ONLY if the blip model is quadratic in treatment. See Details for further clarification.
treat.mod	A list of formula objects specifying the treatment model for each stage in order. Treatment variable should be included as the dependent variable. If treatment is binary `glm(family = binomial)` will be used to obtain parameter estimates; if treatment is multi-nomial, `multinom()` will be used to obtain parameter estimates; and if treatment is continuous, `lm()` will be used.
tf.mod	A list of formula objects specifying covariates of the treatment-free model for each stage in order. No dependent variable should be specified.
data	A data frame containing all necessary covariates and treatments contained in the models. Missing data should be coded as NA.
method	The DTR method to be used, choose "dwols" for dynamic WOLS, "gest" for G-estimation, or "qlearn" for Q-learning.
interactive	If TRUE on-screen prompts will guide the user through the specification of blip, treatment, and treatment-free models.
treat.type	A character object. Must be one of {"bin", "multi", "cont"} indicating that the treatments at each stage are binary, multinomial, or continuous, respectively. Each stage must have the same treatment type.
treat.fam	A character or family object. The description of the dose distribution along with the link function to be used in the treatment model for computing weights; should be specified in a similar format as that used in `glm()`. If character object, must be one of {"gaussian", "Gamma"}, for which `gaussian(link = "identity")` or `Gamma(link = "log")` will be used, respectively. Input is ignored for `treat.type = "bin"` and `treat.type = "multi"`.
weight	The form of the treatment weight. See details.
n.bins	An integer object. The number of bins (levels) to be used for categorizing continuous doses. This input is required only when `treat.type = "cont"` and `weight = "wo"` or `weight = "qpom"`.
treat.range	For continuous treatments. Specify the maximum/minimum value that treatments can be take. If unspecified then the minimum/maximum value of observed treatments is used. If you wish to have unrestricted treatments set this option to `c(-Inf, +Inf)`. If each stage has its own range, provide as a list, the ith element providing the min and max for the ith stage treatment.
treat.wgt.man	NULL or a list of vectors of known treatment weights can be specified to be used instead of hard-coded treatment weight options. The i^{th} element of the list contains the multiplicative weights for the i^{th} stage. Each vector must be of length n, the number of participants. Used only for `method = "dwols"`.

`var.estim` Covariance matrix estimation method, either "bootstrap" or "sandwich" for sandwich estimation.

`full.cov` A logical. If `TRUE`, the full covariance matrix will be returned. If `FALSE`, only the terms pertaining to the blip parameters are returned.

`bootstrap.controls`
A named list specifying control parameters of the bootstrap if `var.estim = "bootstrap"`. Available controls are:

B: The number of bootstrap samples.

M: The subsample size for m out of n bootstrap.

type: The type of bootstrap. Must be one of {"standard", "empirical", "normal"}. The last two are parametric bootstraps.

truncate: A number between 0 and 0.5. The lowest and highest specified proportion of parameter estimates will be replaced by the relevant quantiles affording some robustness to extreme values when estimating covariance.

verbose: If TRUE, estimated time to completion will be printed to the console every ~30 seconds.

interrupt: If TRUE then user will be given the option to abort the bootstrap without error if estimated time to completion exceeds 10 minutes.

`missing` A character object. Must be one of {"drop", "ipw"}. If set to "ipw" and covariate or treatment data are missing then inverse probability weights are used. The complete case probability is estimated via logistic regression. If set to "drop" and data are missing, participants with missing data are dropped for all stage analyses.

`missing.mod` An optional list of formula objects specifying the model for the inverse probability of weights for each stage in order. No dependent variable should be specified. If `missing = "ipw"` and `missing.mod = NULL`, then the models are assumed to be linear comprising the full covariate history derived from all of the previous stage models.

`dtr` A logical object. If `TRUE`, use the DTR estimation approach, which estimates the stage pseudo-outcome by adding a regret function. If `FALSE`, use an 'effect estimation' approach, which treats the observed outcome as being equal to an outcome assuming no treatment is received at any stage, plus a blip component at each stage; each stage pseudo-outcome is generated by subtracting a blip function. Note that most of the DTR-specific output will either be suppressed or irrelevant.

`x` An object of class 'DTRreg'.

`...` Ignored.

`object` An object of class 'DTRreg'.

Details

`DTRreg()` allows the estimation of optimal dynamic treatment regimens (DTRs, also known as adaptive treatment strategies) from multi-stage trials using G-estimation, dynamic weighted ordinary least squares (dWOLS), and genearlized dWOLS. All methods focus on estimating the parameters of the blip: a model of the difference in expected outcome under the observed treatment and

some reference treatment (usually a control) at a given stage, assuming identical histories and optimal treatment thereafter. The reader is referred to Chakraborty and Moodie (2013) for a thorough introduction and review of DTR methods. The dWOLS method may be used to obtain parameter estimates identical to those from Q-learning (by setting `weight = "none"`). This option is intended primarily for exploratory purposes; the authors note that there is a dedicated R package for Q-learning (qLearn), although it is limited to the 2-stage setting; multi-stage settings are available in R package DynTxRegime.

This implementation assumes an outcome regression model of the form E(Y|X=x,A=a) = tf.mod + a blip.mod. That is – the input `blip.mod` formula should include the treatment variable *ONLY* if it is quadratic. For example, if the full blip model is linear in the treatment variable

$$\sim a\psi_0 + ax\psi_1,$$

then the input should model should be `blip.mod = ~ x`. If the full blip model is quadratic in the treatment variable

$$\sim a\psi_0 + a^2\psi_1 + ax\psi_2 + a^2x\psi_3,$$

`blip.mod = ~ a*x`. For continuous treatments, only quadratic blip functions are supported.

All methods require the specification of three models for each stage of the analysis: a treatment model (conditional mean of the treatment variable), a treatment-free model (conditional mean of outcome assuming only reference treatments are used), and a blip model. Only the blip model must be correctly specified (or over-specified), with consistent parameter estimates obtainable if at least one of the other two models is correctly specified. Note that all of these must be specified as lists of formula objects, even if only one stage of treatment is considered.

Note that as is conventional, it is assumed a larger value of the outcome is preferred (which can be easily achieved via transformation of your data if necessary).

When treatment is binary, if confidence intervals are computed (via specification of `var.estim` other than "none"), then DTRreg will calculate the proportion of subjects at each stage for whom optimal treatment is non-unique. If this proportion exceeds 0.05 a non-regularity warning will be displayed, along with the proportion of subjects for whom this is the case. Note that this warning is only displayed if a variance estimation option is selected.

Several treatment weight function options have been implemented within the package:

- "none": No treatment weights applied. If `method = "dWOLS"`, this selection results in the implementation of Q-learning, modified slightly to use the G-estimation or dWOLS style pseudo-outcome (computed using the observed outcome modified by the estimated treatment effect) rather than the traditional Q-learning outcome (predicted based on model only, rather than observed outcome with treatment effect).
- "ipw": weights based on the inverse probability of treatment. For binary treatments, a logistic regression is used. For multinomial, a multinomial log-linear model is fit using `multinom`. For continuous treatments, a GLM with the specified family and link function provided in the `treat.fam` argument is used.
- "cipw": inverse probability of treatment weights as described for "ipw" and capped at the 99th percentile of the observed weights.
- "qpom": weights based on the stabilized inverse probability of treatment applied to the categorized (into n.bins bins) continuous doses or multinomial treatments; probabilities are calculated using a proportional odds model. This weight is appropriate only for continuous and multinomial treatments.

- "wo": overlap weights for the categorized continuous doses or multinomial treatments (Li and Li, 2019). This weight is appropriate only for continuous treatments.
- "abs": Absolute difference $|A - E[A|...]|$. appropriate only for binary treatments.
- "manual": User provides weights through input `treat.wgt.man`. Manual treatments are only used in dwols.

Value

An object of class `DTRreg`, a list including elements

`K:` The number of decision points.

`beta:` A list. The ith element contains the parameter estimates of the ith stage treatment-free model.

`psi:` A list. The ith element contains the parameter estimates of the ith stage blip model.

`covmat:` A list. The ith element contains covariance matrix of the ith stage blip parameter estimates.

`nonreg:` Non-regularity estimates.

`setup:` A list detailing the input parameter settings used for the analysis

models: A list of the models used for the analysis.

method: The parameter estimation method.

var.estim: The variance esetimation method.

cc.modeled: If TRUE, missing data was modeled. If FALSE, cases with missing data were removed from the analysis.

tx.weight: The treatment weighting used for the analysis.

tx.type: Treatment was binary, multinomial, or continuous.

n.bins: The number of bins (levels) used for categorizing continuous doses when `tx.weight = "wo"` or `tx.weight = "qpom"`.

tx.wgt.man: Any user provided treatment weights.

tx.range: For continuous treatments, the range of allowed treatment values.

tx.family: The description of the dose distribution along with the link function used in the continuous treatment model.

boot.controls: A list of the bootstrap controls.

type: The type of effect. Dynamic treatment regime or treatment effect.

`training_data:` A list containing the training data.

data: The covariates and treatment data.

outcome: The outcome of interest.

A: The treatment variables, possibly recoded to adhere to internal code requirements.

`analysis:` A list containing the primary results of each stage analysis.

n: The number of participants included in the stage analysis.

last.stage: The last stage each participant was included in the analysis.

prob.cc: The complete case probabilities.

cc.mod.fitted: The regression objects returned for estimating the complete case probabilities.

cc.wgt: The complete case weights.

cts: The treatment type at each stage.

tx.mod.fitted: The regression objects returned for estimating the treatment probabilities.

A.hat: The estimated or provided treatment probabilities.

tx.wgt: The treatment weights.

outcome.fit: The regression objects returned for each stage outcome regression.

Y: The pseudo-outcomes.

regret: Estimates of the regret for each subject based on observed treatment and blip parameter estimates.

opt.treat: Optimal treatment decisions for each subject at each stage of treatment.

opt.Y: Predicted optimal outcome under recommended regimen.

`call:` The original function call.

The functions `coef()`, `predict()` and `confint()` may be used with such model objects. The first two have specific help files for their implementation, while `confint()` is used in the same way as the standard `confint()` command, with the exception of the `parm` option, which is not available.

Author(s)

Michael Wallace

Shannon T. Holloway

References

Chakraborty, B., Moodie, E. E. M. (2013) *Statistical Methods for Dynamic Treatment Regimes*. New York: Springer.

Robins, J. M. (2004) *Optimal structural nested models for optimal sequential decisions*. In Proceedings of the Second Seattle Symposium on Biostatistics, D. Y. Lin and P. J. Heagerty (eds), 189–326. New York: Springer.

Wallace, M. P., Moodie, E. E. M. (2015) Doubly-Robust Dynamic Treatment Regimen Estimation Via Weighted Least Squares. *Biometrics* **71**(3), 636–644 (doi:10.1111/biom.12306.)

Simoneau, G., Moodie, E. E. M., Nijjar, J. S., and Platt, R. W. (2020) Finite Sample Variance Estimation for Optimal Dynamic Treatment Regimes of Survival Outcomes. *Statistics in Medicine* **39**, 4466-4479.

Efron, B., and Tibshirani, R. (1986) Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy *Source: Statistical Science* **1** 54-75.

Examples

```
data(twoStageCont)

# models to be passed to DTRreg
# blip model
blip.mod <- list(~ X1, ~ X2)
# treatment model (correctly specified)
treat.mod <- list(A1 ~ X1, A2 ~ 1)
# treatment-free model (incorrectly specified)
tf.mod <- list(~ X1, ~ X2)

# perform G-estimation
mod1 <- DTRreg(twoStageCont$Y, blip.mod, treat.mod, tf.mod,
               data = twoStageCont, method = "gest")
mod1
```

DWSurv *DTR estimation and inference for time-to-event data using DWSurv*

Description

Dynamic treatment regimen estimation and inference via dynamic weighted survival modeling (DWSurv). Inference for the blip estimators with single- and multi-stage data.

The function `DWSurv()` allows estimating an optimal dynamic treatment regime from multi-stage trials or observational data when the outcome of interest is survival time subject to right-censoringg. The dynamic weighted survival modeling (DWSurv) algorithm is implemented. The method focuses on estimating the parameters of the blip: a model of the difference in expected outcome under the observed treatment and some reference treatment (usually a control) at a given stage, assuming identical histories and optimal treatment thereafter.

The method requires the specification of four models for each stage of the analysis: a treatment model (conditional mean of the treatment variable), a censoring model, a treatment-free model (conditional mean of outcome assuming only reference treatments are used), and a blip model. Only the blip model must be correctly specified (or over-specified), with consistent parameter estimates obtainable if at least one of the treatment-free or the treatment and censoring models are correctly specified. Note that all of these must be specified as lists of formula objects, even if only one stage of treatment is considered.

Note that as is conventional, it is assumed a larger survival time is preferred (which can be easily achieved via transformation of your data if necessary).

Several treatment weight function options have been implemented within the package:

- "none": No treatment weights applied. If `method = "dWOLS"`, this selection results in the implementation of Q-learning, modified slightly to use the dWOLS style pseudo-outcome (computed using the observed outcome modified by the estimated treatment effect) rather than the traditional Q-learning outcome (predicted based on model only, rather than observed outcome with treatment effect).

- "ipw": weights based on the inverse probability of treatment. For binary treatments, a logistic regression is used. For multinomial, a multinomial log-linear model is fit using multinom. For continuous treatments, a GLM with the specified family and link function provided in the treat.fam argument is used.
- "cipw": inverse probability of treatment weights as described for "ipw" and capped at the 99th percentile of the observed weights.
- "qpom": weights based on the stabilized inverse probability of treatment applied to the categorized (into n.bins bins) continuous doses or multinomial treatments; probabilities are calculated using a proportional odds model. This weight is appropriate only for continuous and multinomial treatments.
- "wo": overlap weights for the categorized continuous doses or multinomial treatments (Li and Li, 2019). This weight is appropriate only for continuous treatments.
- "abs": Absolute difference $|A - E[A|...]|$. appropriate only for binary treatments.
- "manual": User provides treatment weights through input treat.wgt.man.
- "manual.with.censor": User provides combined treatment * censoring weights through input treat.wgt.man. Note that 'cens.mod' should be specified with the event indicator on the right-hand side of the formula (e.g., ~ status).

Usage

```
DWSurv(
  time,
  blip.mod,
  treat.mod,
  tf.mod,
  cens.mod,
  data = NULL,
  method = c("dwols", "qlearn"),
  interactive = FALSE,
  treat.type = c("bin", "multi", "cont"),
  treat.fam = gaussian(link = "identity"),
 weight = c("abs", "ipw", "cipw", "qpom", "wo", "none", "manual", "manual.with.censor"),
  n.bins = 3L,
  treat.range = NULL,
  treat.wgt.man = NULL,
  var.estim = c("none", "bootstrap", "sandwich"),
 bootstrap.controls = list(B = 100L, M = 0L, type = "standard", truncate = 0, verbose =
    FALSE, interrupt = FALSE),
  dtr = TRUE,
  full.cov = FALSE
)
```

Arguments

time	A list of formula specifying the survival time variable for each stage in order. The time variable should be specified on the right hand side of the formula. No dependent variable should be specified. The list should be as long as the number of stages.

`blip.mod` A list of formula objects specifying covariates of the blip function for each stage in order. No dependent variable should be specified. Note that this input should include the treatment variable ONLY if the blip model is quadratic in treatment. See Details for further clarification.

`treat.mod` A list of formula objects specifying the treatment model for each stage in order. Treatment variable should be included as the dependent variable. If treatment is binary `glm(family = binomial)` will be used to obtain parameter estimates; if treatment is multi-nomial, `multinom()` will be used to obtain parameter estimates; and if treatment is continuous, `lm()` will be used.

`tf.mod` A list of formula objects specifying covariates of the treatment-free model for each stage in order. No dependent variable should be specified.

`cens.mod` A list of formula objects specifying the censoring model for each stage in order. The event indicator, which takes value 1 if an event was observed and 0 otherwise, should be included as the dependent variable and should be the same across stages. In the absence of censoring or if censoring weights are provided by the user through 'treat.wgt.man', (i.e., `weight = 'manual.with.censor'`) one still needs to specify an event indicator on the right-hand side of the formula and leave the left-hand side empty (see example below).

`data` A data frame containing all necessary covariates and treatments contained in the models. Missing data should be coded as `NA`.

`method` The DTR method to be used, choose "dwols" for dynamic WOLS, or "qlearn" for Q-learning.

`interactive` If `TRUE` on-screen prompts will guide the user through the specification of blip, treatment, and treatment-free models.

`treat.type` A character object. Must be one of {"bin", "multi", "cont"} indicating that the treatments at each stage are binary, multinomial, or continuous, respectively. Each stage must have the same treatment type.

`treat.fam` A character or family object. The description of the dose distribution along with the link function to be used in the treatment model for computing weights; should be specified in a similar format as that used in `glm()`. If character object, must be one of {"gaussian", "Gamma"}, for which `gaussian(link = "identity")` or `Gamma(link = "log")` will be used, respectively. Input is ignored for `treat.type = "bin"` and `treat.type = "multi"`.

`weight` The form of the treatment weight. See details.

`n.bins` An integer object. The number of bins (levels) to be used for categorizing continuous doses. This input is required only when `treat.type = "cont"` and `weight = "wo"` or `weight = "qpom"`.

`treat.range` For continuous treatments. Specify the maximum/minimum value that treatments can be take. If unspecified then the minimum/maximum value of observed treatments is used. If you wish to have unrestricted treatments set this option to `c(-Inf, +Inf)`. If each stage has its own range, provide as a list, the ith element providing the min and max for the ith stage treatment.

`treat.wgt.man` NULL or a list of vectors of known treatment (or treatment * censoring) weights can be specified to be used instead of hard-coded treatment weight options. The i^{th} element of the list contains the multiplicative weights for the i^{th} stage. Each

vector must be of length n, the number of participants. Used only for `method = "dwols"`. If providing the treatment * censoring weights, `cens.mod = NA` must be used.

`var.estim` Covariance matrix estimation method, either "bootstrap" or "sandwich" for sandwich estimation.

`bootstrap.controls`
A named list specifying control parameters of the bootstrap if `var.estim = "bootstrap"`. Available controls are:

B: The number of bootstrap samples.

M: The subsample size for m out of n bootstrap.

type: The type of bootstrap. Must be one of {"standard", "empirical", "normal"}. The last two are parametric bootstraps.

truncate: A number between 0 and 0.5. The lowest and highest specified proportion of parameter estimates will be replaced by the relevant quantiles affording some robustness to extreme values when estimating covariance.

verbose: If TRUE, estimated time to completion will be printed to the console every ~30 seconds.

interrupt: If TRUE then user will be given the option to abort the bootstrap without error if estimated time to completion exceeds 10 minutes.

`dtr` A logical object. If `TRUE`, use the DTR estimation approach, which estimates the stage pseudo-outcome by adding a regret function. If `FALSE`, use an 'effect estimation' approach, which treats the observed outcome as being equal to an outcome assuming no treatment is received at any stage, plus a blip component at each stage; each stage pseudo-outcome is generated by subtracting a blip function. Note that most of the DTR-specific output will either be suppressed or irrelevant.

`full.cov` A logical. If `TRUE`, the full covariance matrix will be returned. If `FALSE`, only the terms pertaining to the blip parameters are returned.

Value

An object of class `DWSurv`, a list including elements

`K:` The number of decision points.

`beta:` A list. The ith element contains the parameter estimates of the ith stage treatment-free model.

`psi:` A list. The ith element contains the parameter estimates of the ith stage blip model.

`covmat:` A list. The ith element contains covariance matrix of the ith stage blip parameter estimates.

`nonreg:` Non-regularity estimates.

`setup:` A list detailing the input parameter settings used for the analysis

- models: A list of the models used for the analysis.
- method: The parameter estimation method.

- var.estim: The variance estimation method.
- cc.modeled: If TRUE, missing data was modeled. If FALSE, cases with missing data were removed from the analysis.
- tx.weight: The treatment weighting used for the analysis.
- tx.type: Treatment was binary, multinomial, or continuous.
- n.bins: The number of bins (levels) used for categorizing continuous doses when `tx.weight = "wo"` or `tx.weight = "qpom"`.
- tx.wgt.man: Any user provided treatment weights.
- tx.range: For continuous treatments, the range of allowed treatment values.
- tx.family: The description of the dose distribution along with the link function used in the continuous treatment model.
- boot.controls: A list of the bootstrap controls.
- type: The type of effect. Dynamic treatment regime or treatment effect.

`training_data:` A list containing the training data.

- data: The covariates and treatment data.
- outcome: The outcome of interest.
- A: The treatment variables, possibly recoded to adhere to internal code requirements.

`analysis:` A list containing the primary results of each stage analysis.

- n: The number of participants included in the stage analysis.
- last.stage: The last stage each participant was included in the analysis.
- prob.cens: The complete case probabilities.
- cens.mod.fitted: The regression objects returned for estimating the complete case probabilities.
- cens.wgt: The complete case weights.
- cts: The treatment type at each stage.
- tx.mod.fitted: The regression objects returned for estimating the treatment probabilities.
- A.hat: The estimated or provided treatment probabilities.
- tx.wgt: The treatment weights.
- outcome.fit: The regression objects returned for each stage outcome regression.
- Y: The pseudo-outcomes.
- regret: Estimates of the regret for each subject based on observed treatment and blip parameter estimates.
- opt.treat: Optimal treatment decisions for each subject at each stage of treatment.
- opt.Y: Predicted optimal outcome under recommended regimen.

`call:` The original function call.

The functions `coef()`, `predict()` and `confint()` may be used with such model objects. The first two have specific help files for their implementation, while `confint()` is used in the same way as the standard `confint()` command, with the exception of the `parm` option, which is not available.

References

Simoneau, G., Moodie, E. E. M., Wallace, M.P., Platt, R. W. (2020) Optimal Dynamic Treatment Regimes with Survival Endpoints: Introducing DWSurv in the R package DTRreg. *Journal of Statistical Computation and Simulation*. **90**, 2991-3008. (doi:10.1080/00949655.2020.1793341)

Simoneau, G., Moodie, E. E. M., Nijjar, J. S., Platt, R. W. (2019) Estimating Optimal Dynamic Treatment with Survival Outcomes. *Journal of the American Statistical Association*, **115**, 1531-1539 (doi:10.1080/01621459.2019.1629939).

Wallace, M. P., Moodie, E. E. M., Stephens, D. A. (2017) Dynamic Treatment Regimen Estimation via Regression-Based Techniques: Introducing R Package DTRreg. *Journal of Statistical Software* **80**(2), 1–20 (doi:10.18637/jss.v080.i02).

Simoneau, G., Moodie, E. E. M., Nijjar, J. S., and Platt, R. W. (2020) Finite Sample Variance Estimation for Optimal Dynamic Treatment Regimes of Survival Outcomes. *Statistics in Medicine* **39**, 4466-4479.

Efron, B., and Tibshirani, R. (1986) Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy *Source: Statistical Science* **1** 54-75.

Examples

```
#### example single run of a 2-stage DWSurv analysis
data(twoStageCens)
mod <- DWSurv(time = list(~ T1, ~ T2),
              blip.mod = list(~ X11, ~ X21),
              treat.mod = list(A1 ~ X11, A2 ~ 1),
              tf.mod = list(~ X11 + X12, ~ X21 + X22 + X11),
              cens.mod = list(delta ~ 1, delta ~ X11),
              var.estim = "sandwich",
              data = twoStageCens)
mod

#### example in the absence of censoring
data(twoStageSurv)
mod_nocensoring <- DWSurv(time = list(~ T1, ~ T2),
                          blip.mod = list(~ X11, ~ X21),
                          treat.mod = list(A1 ~ X11, A2 ~ 1),
                          tf.mod = list(~ X11 + X12, ~ X21 + X22 + X11),
                          cens.mod = list(~ delta, ~ delta),
                          var.estim = "sandwich",
                          data = twoStageSurv)
mod_nocensoring
```

`plot.DTRreg` *Diagnostic Plots for DTR Estimation*

Description

Diagnostic plots for assessment of treatment, treatment-free, and blip models following DTR estimation using DTRreg or DWSurv.

DTR estimation using G-estimation and dWOLS requires the specification of three models: the treatment, treatment-free, and blip. The treatment model may be assessed via standard diagnostics, whereas the treatment-free and blip models may be simultaneously assessed using diagnostic plots introduced by Rich et al. The plot() function first presents diagnostic plots that assess the latter, plotting fitted values against residuals and covariates following DTR estimation. If there is any evidence of a relationship between the variables in these plots, this is evidence that at least one of the blip or treatment-free models is mis-specified.

Following these plots, the plot() function will present standard diagnostic plots for the treatment model. These are produced directly by the standard plot() command applied to the models that were fit. For example, if treatment is binary, the resulting plots are the same as those that are generated by the plot() command applied to a glm object for logistic regression.

Usage

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

Arguments

x	A model object generated by the functions DTRreg and DWSurv.
...	Space for additional arguments (not currently used)

Author(s)

Michael Wallace

References

Chakraborty, B., Moodie, E. E. M. (2013) *Statistical Methods for Dynamic Treatment Regimes*. New York: Springer.

Rich B., Moodie E. E. M., Stephens D. A., Platt R. W. (2010) Model Checking with Residuals for G-estimation of Optimal Dynamic Treatment Regimes. *International Journal of Biostatistics* **6**(2), Article 12.

Robins, J. M. (2004) *Optimal structural nested models for optimal sequential decisions*. In Proceedings of the Second Seattle Symposium on Biostatistics, D. Y. Lin and P. J. Heagerty (eds), 189-326. New York: Springer.

Wallace, M. P., Moodie, E. M. (2015) Doubly-Robust Dynamic Treatment Regimen Estimation Via Weighted Least Squares. *Biometrics* **71**(3), 636-644 (doi:10.1111/biom.12306.)

Examples

```
# example single run of a 2-stage g-estimation analysis

set.seed(1)
```

```
# expit function
expit <- function(x) { 1.0 / (1.0 + exp(-x)) }

# sample size
n <- 10000

# variables (X = patient information, A = treatment)
X1 <- rnorm(n)
A1 <- rbinom(n, 1, expit(X1))
X2 <- rnorm(n)
A2 <- rbinom(n, 1, expit(X2))

# blip functions
gamma1 <- A1 * (1 + X1)
gamma2 <- A2 * (1 + X2)

# observed outcome: treatment-free outcome plus blip functions
Y <- exp(X1) + exp(X2) + gamma1 + gamma2 + rnorm(n)

# models to be passed to DTRreg
# blip model
blip.mod <- list(~ X1, ~ X2)
# treatment model (correctly specified)
treat.mod <- list(A1 ~ X1, A2 ~ 1)
# treatment-free model (incorrectly specified)
tf.mod <- list(~ X1, ~ X2)

# perform G-estimation
mod1 <- DTRreg(twoStageCont$Y, blip.mod, treat.mod, tf.mod,
               data = twoStageCont, method = "gest")

# model diagnostics: note treatment-free model is mis-specified
plot(mod1)
```

`predict.DTRreg` *Optimal Outcome Prediction for DTRs*

Description

Predicted outcome assuming optimal treatment (according to analysis via G-estimation or dWOLS) was followed. Assumes blip and treatment-free models correctly specified.

This function may be used in a similar fashion to more traditional modeling commands (such as lm). Users are referred to the primary 'DTRreg()' and 'DTRSurv()' help command (and associated literature) for information concerning model specification. In particular, we note that the predict function assumes that the treatment-free model has been correctly specified, as the treatment-free parameters are used in the prediction process.

Usage

```
## S3 method for class 'DTRreg'
predict(object, newdata, treat.range = NULL, ...)
```

Arguments

object	A model object generated by the function 'DTRreg()' or 'DWSurv()'.
newdata	A dataset (usually the data analyzed by DTRreg for which predicted outcomes are desired. If a new dataset is provided, variable names should correspond to those presented to 'DTRreg()' or 'DWSurv()'.
treat.range	If treatment is continuous (rather than binary), a vectors of the form c(min,max) which specify the minimum and maximum value the treatment may take at stage 1. If unspecified, this will be inferred from the treat.range provided with use of the original DTRreg command. As such, if no treatment range was specified there either, treat.range will be the minimum and maximum observed first stage treatment.
...	Space for additional arguments (not currently used)

Value

An n x 1 matrix of predicted outcome values.

Author(s)

Michael Wallace

References

Chakraborty, B., Moodie, E. E. M. (2013) *Statistical Methods for Dynamic Treatment Regimes*. New York: Springer.

Robins, J. M. (2004) *Optimal structural nested models for optimal sequential decisions*. In Proceedings of the Second Seattle Symposium on Biostatistics, D. Y. Lin and P. J. Heagerty (eds), 189-326. New York: Springer.

Wallace, M. P., Moodie, E. M. (2015) Doubly-Robust Dynamic Treatment Regimen Estimation Via Weighted Least Squares. *Biometrics* **71**(3), 636-644 (doi:10.1111/biom.12306.)

Examples

```
# example single run of a 2-stage g-estimation analysis

set.seed(1)

# expit function
expit <- function(x) { 1.0 / (1.0 + exp(-x)) }

# sample size
n <- 10000
```

```
# variables (X = patient information, A = treatment)
X1 <- rnorm(n)
A1 <- rbinom(n, 1, expit(X1))
X2 <- rnorm(n)
A2 <- rbinom(n, 1, expit(X2))

# blip functions
gamma1 <- A1 * (1 + X1)
gamma2 <- A2 * (1 + X2)

# observed outcome: treatment-free outcome plus blip functions
Y <- exp(X1) + exp(X2) + gamma1 + gamma2 + rnorm(n)

# models to be passed to DTRreg
# blip model
blip.mod <- list(~ X1, ~ X2)
# treatment model (correctly specified)
treat.mod <- list(A1 ~ X1, A2 ~ 1)
# treatment-free model (incorrectly specified)
tf.mod <- list(~ X1, ~ X2)

# perform G-estimation
mod1 <- DTRreg(twoStageCont$Y, blip.mod, treat.mod, tf.mod,
               data = twoStageCont, method = "gest")

# predicted Y for optimal treatment
dat <- data.frame(X1, X2, A1, A2)
predict(mod1, newdata = dat)
```

Index