Package 'ltmle'

November 17, 2016

Type Package

Title Longitudinal Targeted Maximum Likelihood Estimation

Version 0.9-9

Date 2016-11-16

Maintainer Joshua Schwab <joshuaschwab@yahoo.com>

Depends R(>= 3.1.0)

Imports Matrix, matrixStats, speedglm

Suggests SuperLearner, testthat

Description Targeted Maximum Likelihood Estimation (TMLE) of treatment/censoring specific mean outcome or marginal structural model for point-treatment and longitudinal data.

License GPL-2

Collate 'ltmle-package.R' 'GeneralUtilities.R' 'ltmle.R' 'ltmle_sg.R' 'BinaryToCensoring.R' 'DeterministicFunctions.R' 'zzz.R'

URL https://github.com/joshuaschwab/ltmle

BugReports https://github.com/joshuaschwab/ltmle/issues

RoxygenNote 5.0.1

NeedsCompilation no

Author Joshua Schwab [aut, cre],
Samuel Lendle [aut],
Maya Petersen [aut],
Mark van der Laan [aut],
Susan Gruber [ctb]

Repository CRAN

Date/Publication 2016-11-17 08:27:43

R topics documented:

`ltmle-package` *Targeted Maximum Likelihood Estimation for Longitudinal Data*

Description

Targeted Maximum Likelihood Estimation (TMLE) of treatment/censoring specific mean outcome or marginal structural model for point-treatment and longitudinal data. Also provides Inverse Probability of Treatment/Censoring Weighted estimate (IPTW) and maximum likelihood based G-computation estimate (G-comp). Can be used to calculate additive treatment effect, risk ratio, and odds ratio.

Author(s)

Joshua Schwab, Samuel Lendle, Maya Petersen, and Mark van der Laan, with contributions from Susan Gruber

Maintainer: Joshua Schwab <joshuaschwab@yahoo.com>

See Also

`ltmle` #' @references Bang, Heejung, and James M. Robins. "Doubly robust estimation in missing data and causal inference models." Biometrics 61.4 (2005): 962-973.

Lendle, Samuel, Schwab, Joshua, Petersen, Maya and and van der Laan, Mark J "ltmle: An R Package Implementing Targeted Minimum Loss-based Estimation for Longitudinal Data", Forthcoming

Petersen, Maya, Schwab, Joshua and van der Laan, Mark J, "Targeted Maximum Likelihood Estimation of Marginal Structural Working Models for Dynamic Treatments Time-Dependent Outcomes", Journal of Causal Inference, 2014 http://www.ncbi.nlm.nih.gov/pubmed/25909047

Robins JM, Sued M, Lei-Gomez Q, Rotnitsky A. (2007). Comment: Performance of double-robust estimators when Inverse Probability weights are highly variable. Statistical Science 22(4):544-559.

van der Laan, Mark J. and Gruber, Susan, "Targeted Minimum Loss Based Estimation of an Intervention Specific Mean Outcome" (August 2011). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 290. http://biostats.bepress.com/ucbbiostat/paper290

van der Laan, Mark J. and Rose, Sherri, "Targeted Learning: Causal Inference for Observational and Experimental Data" New York: Springer, 2011.

Examples

```
## For examples see examples(ltmle)
```

BinaryToCensoring *BinaryToCensoring*

Description

Helper function for creating censoring columns as factors.

Usage

```
BinaryToCensoring(is.censored, is.uncensored)
```

Arguments

is.censored	binary vector: 0=uncensored, 1=censored
is.uncensored	binary vector: 0=censored, 1=uncensored

Details

Exactly one of `is.censored` and `is.uncensored` must be specified as a *named* argument. All elements of the input vector must be 0, 1, or `NA`

Value

an object of class "`factor`" with levels "censored" and "uncensored"

Author(s)

Joshua Schwab <joshuaschwab@yahoo.com>

See Also

`factor`

Examples

```
 BinaryToCensoring(is.censored=c(0, 1, 1, 0, NA))
 BinaryToCensoring(is.uncensored=c(1, 0, 0, 1, NA))   #the same

 ## Not run:
 BinaryToCensoring(c(0, 1))   #error because the input must be named

## End(Not run)
```

deterministic.g.function_template *Deterministic g/Q functions - examples and templates*

Description

Template for the deterministic.g.function argument to ltmle or ltmleMSM.

Usage

```
deterministic.g.function_template(data, current.node, nodes)

deterministic.Q.function_template(data, current.node, nodes,
  called.from.estimate.g)
```

Arguments

data	the 'data' data.frame passed to ltmle or ltmleMSM
current.node	the column index of data corresponding to the A or C node (for g) or L or Y node (for Q)
nodes	list of column indicies, components: • A Anodes (treatment) • C Cnodes (censoring) • L Lnodes (time-varying covariates) • Y Ynodes (events) • AC Anodes and Cnodes combined and sorted • LY Lnodes and Ynodes combined, sorted, "blocks" removed - see ltmle
called.from.estimate.g	TRUE or FALSE - your function will be called with called.from.estimate.g=TRUE during estimation of g and called.from.estimate.g=FALSE during estimation of Q.

Details

MaintainTreatment and MaintainControl are two commonly used deterministic.g.functions.

The intended use of the templates is for the user to copy and paste the function arguments and body and then fill in the required sections. They will not run as-is. Note that there are no comments in the functions as saved. Versions with comments may be found in Examples section below.

MaintainTreatment and MaintainControl may be passed as-is for the deterministic.g.function argument to ltmle or ltmleMSM

Note that censoring nodes in data may be passed as binaries but they are converted to the preferred format of factors with levels "censored" and "uncensored" before deterministic functions are called. Also note that nodes may be passed to ltmle as either the names of nodes or numerical column indicies, but they are all converted to numerical indicies before deterministic functions are called. If the

survivalFunction argument to ltmle or ltmleMSM is TRUE, the package automatically assumes that once Y jumps to 1, all future Y nodes stay 1 and treatment does not change. It is not necessary to specify this in deterministic functions.

Value

A deterministic.g.function should return a list with components:

is.deterministic
: vector of logicals, length=nrow(data)

prob1
: the probability that data[is.deterministic, current.node] == 1, vector of length 1 or length(which(is.deterministic))

A deterministic.Q.function should return a list with components:

is.deterministic
: vector of logicals, length=nrow(data)

Q.value
: the iterated expectation of the final Y, vector of length 1 or length(which(is.deterministic))

NOTE: The Q.value component is not used or required when called.from.estimate.g is TRUE

Functions

- deterministic.Q.function_template: Template for the deterministic.Q.function argument to ltmle or ltmleMSM.

Author(s)

Joshua Schwab <joshuaschwab@yahoo.com>

See Also

ltmle, ltmleMSM

Examples

```
# Show template for a deterministic.g.function (comments will not be
# shown, see below for comments)
deterministic.g.function_template

# Show template for a deterministic.Q.function (comments will not be
# shown, see below for comments)
deterministic.Q.function_template

# Use MaintainTreatment
set.seed(1)
rexpit <- function(x) rbinom(n = length(x), size = 1, prob = plogis(x))
n <- 100
W <- rnorm(n)
A1 <- rexpit(W)
A2 <- as.numeric(rexpit(W) | A1)  #treatment at time 1 implies treatment at time 2
```

```
Y <- rexpit(W + A1 + A2 + rnorm(n))
data <- data.frame(W, A1, A2, Y)

result <- ltmle(data, Anodes = c("A1", "A2"), Ynodes = "Y", abar = c(1, 1),
    deterministic.g.function = MaintainTreatment)

# deterministic.g.function_template with comments:

deterministic.g.function_template <- function(data, current.node, nodes) {
    # data: the 'data' data.frame passed to ltmle/ltmleMSM current.node: the
    # column index of data corresponding to the A or C node (see
    # is.deterministic below) nodes: list of column indicies, components: A,
    # C, L, Y, AC (Anodes and Cnodes combined and sorted), LY (Lnodes and
    # Ynodes combined, sorted, 'blocks' removed - see ?ltmle) Note that nodes
    # may be passed to ltmle as either the names of nodes or numerical column
    # indicies, but they are all converted to numerical indicies before
    # deterministic.g.function is called

    # deterministic.g.function will be called at all Anodes and Cnodes
    # return(NULL) is equivalent to return(list(is.deterministic=rep(FALSE,
    # nrow(data)), prob1=numeric(0)))

    # define is.deterministic here: vector of logicals, length=nrow(data)
    # define prob1 here: the probability that data[is.deterministic,
    # current.node] == 1, vector of length 1 or
    # length(which(is.deterministic))
    is.deterministic <- stop("replace me!")
    prob1 <- stop("replace me!")
    return(list(is.deterministic = is.deterministic, prob1 = prob1))
}

# deterministic.Q.function_template with comments:

deterministic.Q.function_template <- function(data, current.node, nodes,
    called.from.estimate.g) {
    # data: the 'data' data.frame passed to ltmle/ltmleMSM current.node: the
    # column index of data corresponding to the A or C node (see
    # is.deterministic below) nodes: list of column indicies, components: A,
    # C, L, Y, AC (Anodes and Cnodes combined and sorted), LY (Lnodes and
    # Ynodes combined, sorted, 'blocks' removed - see ?ltmle)
    # called.from.estimate.g: TRUE or FALSE - your function will be called
    # with called.from.estimate.g=TRUE during estimation of g and
    # called.from.estimate.g=FALSE during estimation of Q. During estimation
    # of g, only the is.deterministic element of the return list will be
    # used.  Note that nodes may be passed to ltmle as either the names of
    # nodes or numerical column indicies, but they are all converted to
    # numerical indicies before deterministic.Q.function is called

    # It is not necessary to specify that deterministic Y events (Y==1)
    # indicate a deterministic Q value of 1; this is automatic
    # if the survivalFunction input to ltmle/ltmleMSM is TRUE.
    # deterministic.Q.function will be called at all Lnodes and Ynodes (after
    # removing 'blocks') and Anodes and Cnodes (see called.from.estimate.g
```

```
    # above) return(NULL) is equivalent to
    # return(list(is.deterministic=rep(FALSE, nrow(data)),
    # Q.value=numeric(0)))

    # define is.deterministic here: vector of logicals, length=nrow(data)
    # define Q.value here: the iterated expectation of the final Y, vector of
    # length 1 or length(which(is.deterministic))
    is.deterministic <- stop("replace me!")
    Q.value <- stop("replace me!")
    return(list(is.deterministic = is.deterministic, Q.value = Q.value))
}
```

`ltmle` *Longitudinal Targeted Maximum Likelihood Estimation*

Description

`ltmle` is Targeted Maximum Likelihood Estimation (TMLE) of treatment/censoring specific mean outcome for point-treatment and longitudinal data. `ltmleMSM` adds Marginal Structural Models. Both always provide Inverse Probability of Treatment/Censoring Weighted estimate (IPTW) as well. Maximum likelihood based G-computation estimate (G-comp) can be obtained instead of TMLE. `ltmle` can be used to calculate additive treatment effect, risk ratio, and odds ratio.

Usage

```
ltmle(data, Anodes, Cnodes = NULL, Lnodes = NULL, Ynodes,
  survivalOutcome = NULL, Qform = NULL, gform = NULL, abar, rule = NULL,
  gbounds = c(0.01, 1), Yrange = NULL, deterministic.g.function = NULL,
  stratify = FALSE, SL.library = NULL, estimate.time = TRUE,
  gcomp = FALSE, iptw.only = FALSE, deterministic.Q.function = NULL,
  variance.method = "tmle", observation.weights = NULL, id = NULL)

ltmleMSM(data, Anodes, Cnodes = NULL, Lnodes = NULL, Ynodes,
  survivalOutcome = NULL, Qform = NULL, gform = NULL, gbounds = c(0.01,
  1), Yrange = NULL, deterministic.g.function = NULL, SL.library = NULL,
  regimes, working.msm, summary.measures, final.Ynodes = NULL,
  stratify = FALSE, msm.weights = "empirical", estimate.time = TRUE,
  gcomp = FALSE, iptw.only = FALSE, deterministic.Q.function = NULL,
  variance.method = "tmle", observation.weights = NULL, id = NULL)
```

Arguments

`data`	data frame following the time-ordering of the nodes. See 'Details'.
`Anodes`	column names or indicies in `data` of treatment nodes
`Cnodes`	column names or indicies in `data` of censoring nodes
`Lnodes`	column names or indicies in `data` of time-dependent covariate nodes

`Ynodes` column names or indicies in `data` of outcome nodes

`survivalOutcome`
If `TRUE`, then Y nodes are indicators of an event, and if Y at some time point is 1, then all following should be 1. Required to be `TRUE` or `FALSE` if outcomes are binary and there are multiple Ynodes.

`Qform` character vector of regression formulas for Q. See 'Details'.

`gform` character vector of regression formulas for g or a matrix/array of prob(A=1). See 'Details'.

`abar` binary vector (numAnodes x 1) or matrix (n x numAnodes) of counterfactual treatment or a list of length 2. See 'Details'.

`rule` a function to be applied to each row (a named vector) of `data` that returns a numeric vector of length numAnodes. See 'Details'.

`gbounds` lower and upper bounds on estimated cumulative probabilities for g-factors. Vector of length 2, order unimportant.

`Yrange` NULL or a numerical vector where the min and max of `Yrange` specify the range of all Y nodes. See 'Details'.

`deterministic.g.function`
optional information on A and C nodes that are given deterministically. See 'Details'. Default `NULL` indicates no deterministic links.

`stratify` if `TRUE` stratify on following `abar` when estimating Q and g. If `FALSE`, pool over `abar`.

`SL.library` optional character vector of libraries to pass to `SuperLearner`. `NULL` indicates glm should be called instead of `SuperLearner`. '`default`' indicates a standard set of libraries. May be separately specified for Q and g. See 'Details'.

`estimate.time` if `TRUE`, run an initial estimate using only 50 observations and use this to print a very rough estimate of the total time to completion. No action if there are fewer than 50 observations.

`gcomp` if `TRUE`, run the maximum likelihood based G-computation estimate *instead* of TMLE

`iptw.only` by default (`iptw.only = FALSE`), both TMLE and IPTW are run in `ltmle` and `ltmleMSM`. If `iptw.only = TRUE`, only IPTW is run, which is faster.

`deterministic.Q.function`
optional information on Q given deterministically. See 'Details'. Default `NULL` indicates no deterministic links.

`variance.method`
Method for estimating variance of TMLE. One of "ic", "tmle", "iptw". If "tmle", compute both the robust variance estimate using TMLE and the influence curve based variance estimate (use the larger of the two). If "iptw", compute both the robust variance estimate using IPTW and the influence curve based variance estimate (use the larger of the two). If "ic", only compute the influence curve based variance estimate. "ic" is fastest, but may be substantially anti-conservative if there are positivity violations or rare outcomes. "tmle" is slowest but most robust if there are positivity violations or rare outcomes. "iptw" is a compromise between speed and robustness. variance.method="tmle" or "iptw" are not yet available with non-binary outcomes, gcomp=TRUE, stratify=TRUE, deterministic.Q.function, or numeric gform.

observation.weights
: observation (sampling) weights. Vector of length n. If NULL, assumed to be all 1.

id
: Household or subject identifiers. Vector of length n or NULL. Integer, factor, or character recommended, but any type that can be coerced to factor will work. NULL means all distinct ids.

regimes
: binary array: n x numAnodes x numRegimes of counterfactual treatment or a list of 'rule' functions

working.msm
: character formula for the working marginal structural model

summary.measures
: array: num.regimes x num.summary.measures x num.final.Ynodes - measures summarizing the regimes that will be used on the right hand side of working.msm (baseline covariates may also be used in the right hand side of working.msm and do not need to be included in summary.measures)

final.Ynodes
: vector subset of Ynodes - used in MSM to pool over a set of outcome nodes

msm.weights
: projection weights for the working MSM. If "empirical", weight by empirical proportions of rows matching each regime for each final.Ynode, with duplicate regimes given zero weight. If NULL, no weights. Or an array of user-supplied weights with dimensions c(n, num.regimes, num.final.Ynodes) or c(num.regimes, num.final.Ynodes).

Details

The estimates returned by ltmle are of a treatment specific mean, $E[Y_{\bar{a}}]$, the mean of the final treatment node, where all treatment nodes, A, are set to $\bar{a}$ (abar) and all censoring nodes C are set to 1 (uncensored). The estimates returned by ltmleMSM are similar but are the parameters in a working marginal structural model.

data should be a data frame where the order of the columns corresponds to the time-ordering of the model.

- in censoring columns (Cnodes): factor with two levels: "censored" and "uncensored". The helper function CensoringToBinary can be used to create these factors.
- in treatment columns (Anodes): 1 = treated, 0 = untreated (must be binary)
- in event columns (Ynodes): If survivalOutcome is TRUE, then Y nodes are treated as indicators of a one-time event. See details for survivalOutocme. If survivalOutcome is FALSE, Y nodes are treated as binary if all values are 0 or 1, and are treated as continuous otherwise. If Y nodes are continuous, they may be automatically scaled. See details for Yrange.
- time-dependent covariate columns (Lnodes): can be any numeric data
- Data in Cnodes, Anodes, Lnodes and Ynodes are not used after (to the right of) censoring (or an event when survivalOutcome==TRUE) and may be coded as NA or any other value.
- Columns in data that are before (to the left of) the first of Cnodes or Anodes are treated as baseline variables, even if they are specified as Lnodes.
- After the first of Cnodes, Anodes, Ynodes, or Lnodes, every column must be in one of Cnodes, Anodes, Ynodes, or Lnodes.

If survivalOutcome is TRUE, all Y values are indicators of an event (e.g. death) at or before the current time, where 1 = event and 0 = no event. The events in Ynodes must be of the form where once Y jumps to 1, Y remains 1 at subsequent nodes.

For continuous outcomes, (survivalOutcome==FALSE and some Y nodes are not 0 or 1,) Y values are truncated at the minimum and maximum of Yrange if specified, and then transformed and scaled to be in [0,1]. That is, transformed to (Y-min(Yrange))/(max(Yrange)-min(Yrange)). If Yrange is NULL, it is set to the range of all Y nodes. In that case, Y nodes are only scaled if any values fall outside of [0,1]. For intervention specific means (ltmle), parameter estimates are transformed back based Yrange.

Qform should be NULL, in which case all parent nodes of each L and Y node will be used as regressors, or a named character vector that can be coerced to class "formula". The length of Qform must be equal to length(Lnodes) + length(Ynodes)** and the names and order of the formulas must be the same as the names and order of the L and Y nodes in data. The left hand side of each formula should be "Q.kplus1". If SL.library is NULL, glm will be called using the elements of Qform. If SL.library is specified, SuperLearner will be called after a design matrix is created using Qform.

** If there is a "block" of L and Y nodes not separated by A or C nodes, only one regression is required at the first L/Y node in a block. You can pass regression formulas for the other L/Y nodes, but they will be ignored (with a message). See example 5.

gform should be NULL, in which case all parent nodes of each L and Y node will be used as regressors, or a character vector that can be coerced to class "formula", or a matrix/array of Prob(A=1). If gform is a character vector, the length of gform must be equal to length(Anodes) + length(Cnodes) and the order of the formulas must be the same as the order the A and C nodes appear in data. The left hand side of each formula should be the name of the Anode or Cnode. If SL.library is NULL, glm will be called using the elements of gform. If SL.library is specified, SuperLearner will be called after a design matrix is created using gform.

In ltmle, gform can also be a n x numACnodes matrix where entry (i, j) is the probability that the ith observation of the jth A/C node is 1 (if an Anode) or uncensored (if a Cnode), conditional on following abar up to that node. In ltmleMSM, gform can similarly be a n x numACnodes x numRegimes array, where entry (i, j, k) is the probability that the ith observation of the jth A/C node is 1 (if an Anode) or uncensored (if a Cnode), conditional on following regime k up to that node. If gform is a matrix/array, deterministic.g.function will not be used and should be NULL.

abar specifies the counterfactual values of the Anodes, using the order they appear in data and should have the same length (if abar is a vector) or number of columns (if abar is a matrix) as Anodes.

rule can be used to specify a dynamic treatment rule. rule is a function applied to each row of data which returns the a numeric vector of the same length as Anodes.

abar and rule cannot both be specified. If one of them if a list of length 2, additive treatment effect, risk ratio, and odds ratio can be computed using summary.ltmleEffectMeasures.

regimes can be a binary array: n x numAnodes x numRegimes of counterfactual treatment or a list of 'rule' functions as described above for the rule parameter for the ltmle function

deterministic.g.function can be a function used to specify model knowledge about value of Anodes and/or Cnodes that are set deterministically. For example, it may be the case that once a patient starts treatment, they always stay on treatment. For details on the form of the function and examples, see deterministic.g.function_template

deterministic.Q.function can be a function used to specify model knowledge about the final event state. For example, it may be the case that a patient can complete the study at some intermediate time point, in which case the probability of death is 0 (assuming they have not died already). For details on the form of the function and examples, see deterministic.Q.function_template

SL.library may be a character vector of libraries (or NULL or 'default'), in which case these libraries are used to estimate both Q and g OR a list with two components, Q and g, where each is a character vector of libraries (or NULL or 'default'). NULL indicates glm should be called instead of SuperLearner If SL.library is the string 'default', SL.library is set to list("SL.glm", "SL.stepAIC", "SL.bayesgl Note that the default set of libraries consists of main terms models. It may be advisable to include squared terms, interaction terms, etc in gform and Qform or include libraries that consider non-linear terms.

If attr(SL.library, "return.fit") == TRUE, then fit$g and fit$Q will return full SuperLearner objects. If not, only a summary matrix will be returned to save memory.

The print method for ltmle objects only prints the tmle estimates.

Value

ltmle returns an object of class "ltmle" (unless abar or rule is a list, in which case it returns an object of class ltmleSummaryMeasures, which has the same components as ltmleMSM.) The function summary (i.e. summary.ltmle) can be used to obtain or print a summary of the results. An object of class "ltmle" is a list containing the following components:

estimates — a named vector of length 4 with elements, each an estimate of $E[Y_{bara}]$:

- tmle - Targeted Maximum Likelihood Estimate [NULL if gcomp is TRUE]
- iptw - Inverse Probability of Treatment/Censoring Weighted estimate
- gcomp - maximum likelihood based G-computation estimate [NULL if gcomp is FALSE]

IC — a list with the following components of Influence Curve values

- tmle - vector of influence curve values for Targeted Maximum Likelihood Estimate [NULL if gcomp is TRUE]
- iptw - vector of influence curve values for Inverse Probability of Treatment/Censoring Weighted estimate
- gcomp - vector of influence curve values for Targeted Maximum Likelihood Estimate without updating [NULL if gcomp is FALSE]

cum.g — cumulative g, after bounding: for ltmle, n x numACnodes, for ltmleMSM, n x numACnodes x num.regimes

cum.g.unbounded — cumulative g, before bounding: for ltmle, n x numACnodes, for ltmleMSM, n x numACnodes x num.regimes

cum.g.used — binary - TRUE if an entry of cum.g was used in the updating step (note: even if cum.g.used is FALSE, a small value of cum.g.unbounded may still indicate a positivity problem): for ltmle, n x numACnodes, for ltmleMSM, n x numACnodes x num.regimes

call — the matched call

gcomp	the gcomp input
formulas	a list with elements Qform and gform
fit	a list with the following components

- g - list of length numACnodes - glm or SuperLearner (see Details) return objects from fitting g regressions
- Q - list of length numLYnodes - glm or SuperLearner (see Details) return objects from fitting Q regressions
- Qstar - list of length numLYnodes - glm (or numerical optimization if glm fails to solve the score equation) return objects from updating the Q fit

ltmleMSM returns an object of class "ltmleMSM" The function summary (i.e. summary.ltmleMSM) can be used to obtain or print a summary of the results. An object of class "ltmleMSM" is a list containing the following components:

beta	parameter estimates for working.msm using TMLE (GCOMP if gcomp input is TRUE)
beta.iptw	parameter estimates for working.msm using IPTW
IC	matrix, n x numBetas - influence curve values for TMLE (without updating if gcomp input is TRUE)
IC.iptw	matrix, n x numBetas - influence curve values for IPTW
msm	object of class glm - the result of fitting the working.msm
cum.g	array, n x numACnodes x numRegimes - cumulative g, after bounding
cum.g.unbounded	array, n x numACnodes x numRegimes - cumulative g, before bounding
call	the matched call
gcomp	the gcomp input
formulas	a list with elements Qform and gform
fit	a list with the following components

- g - list of length numRegimes of list of length numACnodes - glm or SuperLearner (see Details) return objects from fitting g regressions
- Q - list of length numLYnodes - glm or SuperLearner (see Details) return objects from fitting Q regressions
- Qstar - list of length numLYnodes - glm (or numerical optimization if glm fails to solve the score equation) return objects from updating the Q fit

Functions

- ltmleMSM: Longitudinal Targeted Maximum Likelihood Estimation for a Marginal Structural Model

Author(s)

Joshua Schwab <joshuaschwab@yahoo.com>, Samuel Lendle, Maya Petersen, and Mark van der Laan

References

Lendle, Samuel, Schwab, Joshua, Petersen, Maya and and van der Laan, Mark J "ltmle: An R Package Implementing Targeted Minimum Loss-based Estimation for Longitudinal Data", Forthcoming

Petersen, Maya, Schwab, Joshua and van der Laan, Mark J, "Targeted Maximum Likelihood Estimation of Marginal Structural Working Models for Dynamic Treatments Time-Dependent Outcomes", Forthcoming

van der Laan, Mark J. and Gruber, Susan, "Targeted Minimum Loss Based Estimation of an Intervention Specific Mean Outcome" (August 2011). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 290. http://biostats.bepress.com/ucbbiostat/paper290

van der Laan, Mark J. and Rose, Sherri, "Targeted Learning: Causal Inference for Observational and Experimental Data" New York: Springer, 2011.

See Also

summary.ltmle, summary.ltmleMSM, SuperLearner, deterministic.g.function_template, deterministic.Q.functi

Examples

```
rexpit <- function(x) rbinom(n=length(x), size=1, prob=plogis(x))

# Example 1: Single time point example. True value of E[Y_1] (expected value of
#   Y setting A to 1) is approximately 0.5939.
set.seed(2)
n <- 1000
W1 <- rnorm(n)
W2 <- rbinom(n, size=1, prob=0.3)
W3 <- rnorm(n)
A <- rexpit(-1 + 2 * W1^2)
Y <- rexpit(-0.5 + 2 * W1^2 + 0.5 * W2 - 0.5 * A + 0.2 * W3 * A
       - 1.1 * W3 + 0.2 * rnorm(n))

data <- data.frame(W1, W2, W3, A, Y)


 #This takes about 4 seconds to run
library(SuperLearner)

#SuperLearner semiparametric estimation using all parents as regressors
result1 <- ltmle(data, Anodes="A", Lnodes=NULL, Ynodes="Y", abar=1,
  SL.library=c("SL.glm", "SL.step", "SL.mean"))
summary(result1)
summary(result1, estimator="iptw")

#SuperLearner semiparametric estimation using correctly specified regressors
result1a <- ltmle(data, Anodes="A", Lnodes=NULL, Ynodes="Y",
 Qform=c(Y="Q.kplus1 ~ I(W1^2) + W2 + W3*A"), gform="A ~ I(W1^2)", abar=1,
 SL.library=c("SL.glm", "SL.step", "SL.mean"))
summary(result1a)
```

```
#glm using correctly specified Qform and gform
result.abar1 <- ltmle(data, Anodes="A", Lnodes=NULL, Ynodes="Y",
 Qform=c(Y="Q.kplus1 ~ I(W1^2) + W2 + W3*A"), gform="A ~ I(W1^2)",
 abar=1, SL.library=NULL)

 #This takes about 18 seconds to run
#Get summary measures (additive treatment effect, odds ratio, relative risk)
#  for abar=1 vs abar=0
result.compare <- ltmle(data, Anodes="A", Lnodes=NULL, Ynodes="Y",
                      Qform=c(Y="Q.kplus1 ~ I(W1^2) + W2 + W3*A"), gform="A ~ I(W1^2)",
                      abar=list(1, 0), SL.library=NULL)
summary(result.compare)


# Example 2: Longitudinal example. Includes informative censoring and treatment.
# Time ordering of data is W, C1, L1, A1, Y1, C2, L2, A2, Y2
# True value of E[Y_(1,1,1,1)] (expected value of Y setting C1, A1, C2, A2 all to 1)
#  is approximately 0.413.
# A1 is known to always be 1 if L1 < -2, and is 1 with probability 0.1 if L1 > 2
# A2 is known to always be 1 if A1 is 1
# We incorporate this knowledge using deterministic.g.function

# Generate data:
set.seed(2)
ua <- rep(TRUE, n)   #ua = uncensored and alive
L1 <- A1 <- Y1 <- C2.binary <- L2 <- A2 <- Y2 <- as.numeric(rep(NA, n))
W <- rnorm(n)
C1 <- BinaryToCensoring(is.uncensored=rexpit(2 + W))
ua <- ua & C1 == "uncensored"
L1[ua] <- rnorm(n)[ua] + W[ua]
A1[ua] <- rexpit(L1[ua])
A1[ua & L1 < -2] <- 1
A1[ua & L1 >  2] <- rbinom(n, size=1, prob=0.1)[ua & L1 >  2]
Y1[ua] <- rexpit((W + L1 - A1)[ua])
ua <- ua & !Y1
C2.binary[ua] <- rexpit((1 + 0.7 * L1 - A1)[ua])
C2 <- BinaryToCensoring(is.uncensored=C2.binary)
ua <- ua & C2 == "uncensored"
L2[ua] <- (0.5 * L1 - 0.9 * A1 + rnorm(n))[ua]
A2[ua] <- rexpit((0.5 * L1 + 0.8 * L2)[ua]) | A1[ua]
Y2[ua] <- rexpit((0.7 * L1 + L2 - 0.8 * A1 - A2)[ua])
Y2[Y1 == 1] <- 1  # if a patient dies at time 1, record death at time 2 as well
data <- data.frame(W, C1, L1, A1, Y1, C2, L2, A2, Y2)

deterministic.g.function <- function(data, current.node, nodes) {
  if (names(data)[current.node] == "A1") {
    det <- (data$L1 < -2 | data$L1 > 2) & !is.na(data$L1)
    prob1 <- ((data$L1 < -2) * 1 + (data$L1 > 2) * 0.1)[det]
  } else if (names(data)[current.node] == "A2") {
    det <- data$A1 == 1 & !is.na(data$A1)
    prob1 <- 1
  } else if (names(data[current.node]) %in% c("C1", "C2")){
```

```
    return(NULL)  #this returns the default of no deterministic links
   #note that it is not necessary to specify that prior censoring indicates future censoring
  } else {
    stop("unexpected current.node")
  }
  return(list(is.deterministic=det, prob1=prob1))
}

result2 <- ltmle(data, Anodes=c("A1","A2"), Cnodes=c("C1", "C2"),
                Lnodes=c("L1", "L2"), Ynodes=c("Y1", "Y2"), abar=c(1, 1),
                deterministic.g.function=deterministic.g.function, survivalOutcome=TRUE)
summary(result2)

# Example 3: Dynamic treatment, observation weights
# W -> A1 -> L -> A2 -> Y
# Treatment regime of interest is: Always treat at time 1 (A1 = 1),
#   treat at at time 2 (A2 = 1), iff L > 0
# Weight by pmax(W + 2, 0)

set.seed(2)
n <- 1000
W <- rnorm(n)
A1 <- rexpit(W)
L <- 0.3 * W + 0.2 * A1 + rnorm(n)
A2 <- rexpit(W + A1 + L)
Y <- rexpit(W - 0.6 * A1 + L - 0.8 * A2)
data <- data.frame(W, A1, L, A2, Y)

abar <- matrix(nrow=n, ncol=2)
abar[, 1] <- 1
abar[, 2] <- L > 0

result3 <- ltmle(data, Anodes=c("A1", "A2"), Lnodes="L", Ynodes="Y",
  survivalOutcome=TRUE, abar=abar, observation.weights = pmax(W + 2, 0))
summary(result3)

# Example 3.1: The regime can also be specified as a rule function

rule <- function(row) c(1, row["L"] > 0)

result.rule <- ltmle(data, Anodes=c("A1", "A2"), Lnodes="L", Ynodes="Y",
  survivalOutcome=TRUE, rule=rule, observation.weights = pmax(W + 2, 0))
# This should be the same as the above result
summary(result.rule)

# Example 4: Deterministic Q function
# W -> A1 -> Y1 -> L2 -> A2 -> Y2
set.seed(2)
n <- 200
L2 <- A2 <- Y2 <- as.numeric(rep(NA, n))
W <- rnorm(n)
A1 <- rexpit(W)
Y1 <- rexpit(W - A1)
```

```
alive <- Y1 == 0
L2[alive] <- (0.5 * W - 0.9 * A1 + rnorm(n))[alive]
completed.study <- alive & L2 > 0

#Specify that Q is deterministically 0 when L2 is in the history of the
# current Q regression and L2 > 0
#Note 1: det.Q.fun doesn't condition on called.from.estimate.g so g will also be set
#        deterministically after L2 > 0
#Note 2: It is not necessary to specify that Q is deterministically 1 if Y1 is 1; this is automatic
det.Q.fun.4a <- function(data, current.node, nodes, called.from.estimate.g) {
  L2.index <- which(names(data) == "L2")
  stopifnot(length(L2.index) == 1)
  L2.in.history <- L2.index < current.node
  if (! L2.in.history) return(NULL)

  is.deterministic <- data$L2 > 0 & !is.na(data$L2)
  return(list(is.deterministic=is.deterministic, Q.value=0))
}

#patients don't change treatment after leaving study; leave their A2 as NA
A2[alive & !completed.study] <- rexpit((0.5 * W + 0.8 * L2)[alive & !completed.study])

Y2[alive & !completed.study] <- rexpit((L2 - 0.8 * A1 - A2)[alive & !completed.study])
Y2[alive & completed.study] <- 0
Y2[!alive] <- 1  # if a patient dies at time 1, record death at time 2 as well
data <- data.frame(W, A1, Y1, L2, A2, Y2)

result4a <- ltmle(data, Anodes=c("A1","A2"), Lnodes="L2", Ynodes=c("Y1", "Y2"), abar=c(1, 1),
  SL.library=NULL, estimate.time=FALSE, deterministic.Q.function=det.Q.fun.4a, survivalOutcome=TRUE)
#note: You will get the same result if you pass Lnodes=NULL (see next example)
summary(result4a)

#In this variation, suppose that treatment can still change after a patient leaves the study

det.Q.fun.4b <- function(data, current.node, nodes, called.from.estimate.g) {
  #there is no deterministic information when calculating g - treatment may still change
  if (called.from.estimate.g) return(NULL)

  L2.index <- which(names(data) == "L2")
  stopifnot(length(L2.index) == 1)
  L2.in.history <- L2.index < current.node
  if (! L2.in.history) return(NULL)

  is.deterministic <- data$L2 > 0 & !is.na(data$L2)
  return(list(is.deterministic=is.deterministic, Q.value=0))
}

A2[alive] <- rexpit((0.5 * W + 0.8 * L2)[alive])  #patients can change treatment after leaving study
Y2[alive & !completed.study] <- rexpit((L2 - 0.8 * A1 - A2)[alive & !completed.study])
Y2[alive & completed.study] <- 0
Y2[!alive] <- 1  # if a patient dies at time 1, record death at time 2 as well
data <- data.frame(W, A1, Y1, L2, A2, Y2)
```

```
result4b <- ltmle(data, Anodes=c("A1","A2"), Lnodes="L2", Ynodes=c("Y1", "Y2"), abar=c(1, 1),
 SL.library=NULL, estimate.time=FALSE, deterministic.Q.function=det.Q.fun.4b, survivalOutcome=TRUE)
summary(result4b)

# Example 5: Multiple time-dependent covariates and treatments at each time point,
#            continuous Y values
# age -> gender -> A1 -> L1a -> L1b -> Y1 -> A2 -> L2a -> L2b -> Y2
set.seed(2)
n <- 100
age <- rbinom(n, 1, 0.5)
gender <- rbinom(n, 1, 0.5)
A1 <- rexpit(age + gender)
L1a <- 2*age - 3*gender + 2*A1 + rnorm(n)
L1b <- rexpit(age + 1.5*gender - A1)
Y1 <- plogis(age - gender + L1a + 0.7*L1b + A1 + rnorm(n))
A2 <- rexpit(age + gender + A1 - L1a - L1b)
L2a <- 2*age - 3*gender + 2*A1 + A2 + rnorm(n)
L2b <- rexpit(age + 1.5*gender - A1 - A2)
Y2 <- plogis(age - gender + L1a + L1b + A1 + 1.8*A2 + rnorm(n))
data <- data.frame(age, gender, A1, L1a, L1b, Y1, A2, L2a, L2b, Y2)

#Note that gform is not correctly specified in these examples.

#Also show some different ways of specifying the nodes:

result5a <- ltmle(data, Anodes=c(3, 7), Lnodes=c("L1a", "L1b", "L2a", "L2b"),
 Ynodes=grep("^Y", names(data)), abar=c(1, 0), SL.library=NULL, estimate.time=FALSE,
 survivalOutcome=FALSE, gform=c("A1 ~ gender", "A2 ~ age"))
summary(result5a)

#Usually you would specify a Qform for all of the Lnodes and Ynodes but in this case
# L1a, L1b, Y1 is a "block" of L/Y nodes not separated by Anodes or Cnodes (the same is true for
# L2a, L2b, Y2). Only one regression is required at the first L/Y node in a block. You can pass
# regression formulas for the other L/Y nodes, but they'll be ignored.
result5b <- ltmle(data, Anodes=c(3, 7), Lnodes=c("L1a", "L1b", "L2a", "L2b"),
 Ynodes=grep("^Y", names(data)), abar=c(1, 0), estimate.time=FALSE, survivalOutcome=FALSE,
 gform=c("A1 ~ gender", "A2 ~ age"), Qform=c(L1a="Q.kplus1 ~ 1", L2a="Q.kplus1 ~ 1"))
summary(result5b)


#Gives the same result but prints a message saying some regression formulas will be dropped:
result5c <- ltmle(data, Anodes=c(3, 7), Lnodes=c("L1a", "L1b", "L2a", "L2b"),
 Ynodes=grep("^Y", names(data)), abar=c(1, 0), estimate.time=FALSE, survivalOutcome=FALSE,
 gform=c("A1 ~ gender", "A2 ~ age"), Qform=c(L1a="Q.kplus1 ~ 1", L1b="Q.klus1~A1",
 Y1="Q.kplus1~L1a", L2a="Q.kplus1 ~ 1", L2b="Q.klus1~A1", Y2="Q.kplus1~A2 + gender"))

summary(result5c)


#If there were a Anode or Cnode between L1b and Y1, Y1 would also need a Q regression formula


# Example 6: MSM
```

```
# Given data over 3 time points where A switches to 1 once and then stays 1. We want to know
# how death varies as a function of gender, time and an indicator of whether a patient's
# intended regime was to switch before time.
# Note that working.msm includes time and switch.time, which are columns of
# summary.measures; working.msm also includes male, which is ok because it is a baseline
# covariate (it comes before any A/C/L/Y nodes).
data(sampleDataForLtmleMSM)
Anodes <- grep("^A", names(sampleDataForLtmleMSM$data))
Lnodes <- c("CD4_1", "CD4_2")
Ynodes <- grep("^Y", names(sampleDataForLtmleMSM$data))
msm.weights <- matrix(1:12, nrow=4, ncol=3) #just an example (can also use a 200x3x4 array),
                                    #or NULL (for no weights), or "empirical" (the default)

result6 <- ltmleMSM(sampleDataForLtmleMSM$data, Anodes=Anodes, Lnodes=Lnodes, Ynodes=Ynodes,
                    survivalOutcome=TRUE,
                    regimes=sampleDataForLtmleMSM$regimes,
                summary.measures=sampleDataForLtmleMSM$summary.measures, final.Ynodes=Ynodes,
                    working.msm="Y ~ male + time + I(pmax(time - switch.time, 0))",
                    msm.weights=msm.weights, estimate.time=FALSE)
print(summary(result6))


# Example 6.1: regimes can also be specified as a list of rule functions

regimesList <- list(function(row) c(1,1,1),
                      function(row) c(0,1,1),
                      function(row) c(0,0,1),
                      function(row) c(0,0,0))
result.regList <- ltmleMSM(sampleDataForLtmleMSM$data, Anodes=Anodes, Lnodes=Lnodes, Ynodes=Ynodes,
                    survivalOutcome=TRUE, regimes=regimesList,
                summary.measures=sampleDataForLtmleMSM$summary.measures, final.Ynodes=Ynodes,
                    working.msm="Y ~ male + time + I(pmax(time - switch.time, 0))",
                    msm.weights=msm.weights, estimate.time=FALSE)
# This should be the same as the above result
print(summary(result.regList))


# Example 7: variance estimation
# A simple point treatment problem W, A, Y. But there is a positivity problem -
# for small values of W, Prob(A = 1) is very small.
# The true parameter value, E[Y_1] is approximately 0.697
# The true TMLE standard deviation is approximately 0.064,
# the true IPTW standard deviation is approximately 0.058.
set.seed(2)
n <- 1000
W <- rnorm(n)
A <- rexpit(8 * W)
Y <- rexpit(W + A)
r1 <- ltmle(data.frame(W, A, Y), Anodes="A", Ynodes="Y", abar = 1, estimate.time=FALSE)
r2 <- ltmle(data.frame(W, A, Y), Anodes="A", Ynodes="Y", abar = 1, estimate.time=FALSE,
 variance.method="ic")
r3 <- ltmle(data.frame(W, A, Y), Anodes="A", Ynodes="Y", abar = 1, estimate.time=FALSE,
 variance.method="iptw")
```

```
print(summary(r1))
print(summary(r2))
print(summary(r3))
print(summary(r1, estimator="iptw"))
print(summary(r2, estimator="iptw")) #the same - variance.method only affects TMLE
print(summary(r3, estimator="iptw")) #the same - variance.method only affects TMLE
```

`sampleDataForLtmleMSM` *Sample data, regimes, and summary measures*

Description

Sample data for use with ltmleMSM. Data: n=1000: male age CD4_1 A1 Y1 CD4_2 A2 Y2 CD4_3 A3 Y3 A1..A3 are treatment nodes, Y1..Y3 are death, CD4_1..CD4_3 are time varying covariates. We are interested in static regimes where a patient switches at some time. In summary.measures, switch.time is first time where At is 1 (4 if never switch), time is the horizon.

Format

List with three components: data, regimes, summary.measures

Details

regimes: 200 x 3 x 4 [n x numACnodes x numRegimes] summary.measures: 4 x 2 x 3 [numRegimes x numSummaryMeasures x numFinalYnodes]

Source

simulated data

Examples

```
data(sampleDataForLtmleMSM)
```

summary.ltmle *Get standard error, p-value, and confidence interval for one ltmle object Summarizing results from Longitudinal Targeted Maximum Likelihood Estimation (ltmle)*

Description

These functions are methods for class `ltmle` or `summary.ltmle` objects.

Usage

```
## S3 method for class 'ltmle'
summary(object, estimator = ifelse(object$gcomp, "gcomp",
  "tmle"), ...)

## S3 method for class 'ltmleEffectMeasures'
summary(object, estimator = ifelse(object$gcomp,
  "gcomp", "tmle"), ...)

## S3 method for class 'ltmleMSM'
summary(object, estimator = ifelse(object$gcomp, "gcomp",
  "tmle"), ...)

## S3 method for class 'summary.ltmleMSM'
print(x, digits = max(3, getOption("digits") - 3),
  signif.stars = getOption("show.signif.stars"), ...)

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

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

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

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

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

Arguments

object	an object of class "ltmle" or "ltmleMSM" or "ltmleEffectMeasures", usually a result of a call to ltmle or ltmleMSM.

estimator	character; one of "tmle", "iptw", "gcomp". The estimator for which to get effect measures. "tmle" is valid iff the original ltmle/ltmleMSM call used gcomp=FALSE. "gcomp" is valid iff the original ltmle/ltmleMSM call used gcomp=TRUE
x	an object of class "summary.ltmle" or "summary.ltmleMSM" or "ltmleEffectMeasures", usually a result of a call to summary.ltmle or summary.ltmleMSM.
digits	the number of significant digits to use when printing.
signif.stars	logical. If TRUE, significance stars are printed for each coefficient.
...	further arguments passed to or from other methods.

Details

summary.ltmle returns the parameter value of the estimator, the estimated variance, a 95 percent confidence interval, and a p-value.

summary.ltmleEffectMeasures returns the additive treatment effect for each of the two objects in the abar list passed to ltmle. Relative risk, and odds ratio are also returned, along with the variance, confidence interval, and p-value for each.

summary.ltmleMSM returns a matrix of MSM parameter estimates.

Value

summary.ltmle returns an object of class "summary.ltmle", a list with components

treatment	a list with components summarizing the estimate of object • estimate - the parameter estimate of $E[Y_d]$ • std.dev - estimated standard deviation of parameter • p.value - two-sided p-value • CI - vector of length 2 with 95 percent confidence interval
call	the matched call to ltmle for object
estimator	the estimator input argument
variance.estimate.ratio	ratio of the TMLE based variance estimate to the influence curve based variance estimate

summary.ltmleEffectMeasures returns an object of class "summary.ltmleEffectMeasures", a list with same components as summary.ltmle above, but also includes:

effect.measures	a list with components, each with the same components as treatment in summary.ltmle above • treatment - corresponds to the first in the list abar (or rule) passed to ltmle • control - corresponds to the second in the list abar (or rule) passed to ltmle • ATE - average treatment effect • RR - relative risk • OR - odds ratio

summary.ltmleMSM returns an object of class "summary.ltmleMSM", a matrix with rows for each MSM parameter and columns for the point estimate, standard error, 2.5percent confidence interval, 97.5percent confidence interval, and p-value.

See Also

ltmle, summary

Examples

```
rexpit <- function(x) rbinom(n = length(x), size = 1, prob = plogis(x))

# Compare the expected outcomes under two counterfactual plans: Treatment plan:
# set A1 to 1 if W > 0, set A2 to 1 if W > 1.5, always set A3 to 1 Control plan:
# always set A1, A2, and A3 to 0
W <- rnorm(1000)
A1 <- rexpit(W)
A2 <- rexpit(W + 2 * A1)
A3 <- rexpit(2 * A1 - A2)
Y <- rexpit(W - A1 + 0.5 * A2 + 2 * A3)
data <- data.frame(W, A1, A2, A3, Y)
treatment <- cbind(W > 0, W > 1.5, 1)
control <- matrix(0, nrow = 1000, ncol = 3)
result <- ltmle(data, Anodes = c("A1", "A2", "A3"), Ynodes = "Y", abar = list(treatment,
    control))
print(summary(result))

## For examples of summary.ltmle and summary.ltmleMSM, see example(ltmle)
```

Index