Package: tmle3 (via r-universe)

October 30, 2024

Title The Extensible TMLE Framework

Version 0.2.1

Maintainer Jeremy Coyle <jeremyrcoyle@gmail.com>

Description A general framework supporting the implementation of targeted maximum likelihood estimators (TMLEs) of a diverse range of statistical target parameters through a unified interface. The goal is that the exposed framework be as general as the mathematical framework upon which it draws.

Depends R (>= 3.6.0), graphics

Imports sl3 (>= 1.4.5), delayed, data.table, assertthat, R6, uuid, methods, ggplot2, stats, foreach, mvtnorm, magrittr, stringr, digest

Suggests testthat, knitr, origami (>= 1.0.3), tableone, speedglm, rmarkdown, Rsolnp, nnls, tmle, tmle3shift, future, future.apply, xgboost

Remotes github::tlverse/sl3@fix-tests

Additional_repositories https://tlverse.r-universe.dev

License GPL-3

URL https://tlverse.org/tmle3

BugReports https://github.com/tlverse/tmle3/issues

Encoding UTF-8

LazyData yes

LazyLoad yes

VignetteBuilder knitr

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE, r6 = FALSE)

Repository https://tlverse.r-universe.dev

RemoteUrl https://github.com/tlverse/tmle3

RemoteRef devel

RemoteSha fa2bd55249268d20b9f255d804c9634e5c2555b3

Contents

`all_ancestors` *Helper functions for the NPSEM*

Description

`all_ancestors` returns a list of all_ancestors of the specified node. `time_ordering` attempts to find a time_ordering for the variables.

Usage

```
all_ancestors(node_name, npsem)

time_ordering(npsem)
```

Arguments

`node_name`	the node to search for ancestors of
`npsem`	the NPSEM, defined by a list of `tmle3_Node` objects.

bound *Bound (Truncate) Likelihoods*

Description

Bound (Truncate) Likelihoods

Usage

```
bound(x, bounds)
```

Arguments

x	the likelihood values to bound
bounds	Either a length two vector of c(lower,upper) or a lower bound, where the upper is then 1 - lower

CF_Likelihood *Counterfactual Likelihood*

Description

Represents a counterfactual likelihood where one or more likelihood factors has been replaced with an intervention as specified by `intervention_list`. Inherits from `Likelihood`. Other factors (including their updates) are taken from an underlying `observed_likelihood` estimated from observed data.

Usage

```
make_CF_Likelihood(...)
```

Arguments

...	Passes all arguments to the constructor. See documentation for the Constructor below.

Format

`R6Class` object.

Value

`Likelihood` object

Constructor

```
make_CF_Likelihood(observed_likelihood, intervention_list, ...)
```

`observed_likelihood` Likelihood obect specifying the relevant factors of the observed likelihood

`intervention_list` A list of objects inheriting from LF_base, representing the intervention.

`...` Not currently used.

Fields

`observed_likelihood` Likelihood obect specifying the relevant factors of the observed likelihood

`intervention_list` A list of objects inheriting from LF_base, representing the intervention.

See Also

Other Likelihood objects: LF_base, LF_derived, LF_emp, LF_fit, LF_known, LF_static, LF_targeted, Likelihood, Targeted_Likelihood, define_lf()

---

define_lf *Define a Likelihood Factor*

Description

Define a Likelihood Factor

Usage

```
define_lf(LF_class, ...)
```

Arguments

LF_class	the class of likelihood factor. Should inherit from LF_base
...	arguments that define the likelihood factor. See the constructor for the specified LF_class.

See Also

Other Likelihood objects: CF_Likelihood, LF_base, LF_derived, LF_emp, LF_fit, LF_known, LF_static, LF_targeted, Likelihood, Targeted_Likelihood

define_param *Define a Parameter*

Description

Define a Parameter

Usage

```
define_param(Param_class, ...)
```

Arguments

Param_class	the class of the Parameter. Should inherit from Param_base
...	arguments that define the parameter See the constructor for the specified Parameter.

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_MSM, Param_TSM, Param_base, Param_delta, Param_mean, Param_stratified, Param_survival, tmle3_Fit

delta_param_ATE *PAR = Linear Contrast EY1-EY0*

Description

PAR = Linear Contrast EY1-EY0

Usage

```
delta_param_ATE
```

Format

An object of class list of length 4.

delta_param_OR *Odds Ratio odds(Y1)/odds(Y0)*

Description

Odds Ratio odds(Y1)/odds(Y0)

Usage

```
delta_param_OR
```

Format

An object of class `list` of length 5.

delta_param_PAF *PAF = 1 - (1/RR(EY/E0))*

Description

PAF = 1 - (1/RR(EY/E0))

Usage

```
delta_param_PAF
```

Format

An object of class `list` of length 5.

delta_param_PAR *PAR = Linear Contrast EY-EY0*

Description

PAR = Linear Contrast EY-EY0

Usage

```
delta_param_PAR
```

Format

An object of class `list` of length 4.

delta_param_RR *Risk Ratio EY1/EY0*

Description

Risk Ratio EY1/EY0

Usage

```
delta_param_RR
```

Format

An object of class `list` of length 5.

density_formula *Get and Plot Propensity Scores*

Description

Get and Plot Propensity Scores

Usage

```
density_formula(tmle_task, node = "A")

get_propensity_scores(likelihood, tmle_task, node = "A")

propensity_score_plot(likelihood, tmle_task, node = "A")

propensity_score_table(likelihood, tmle_task, node = "A")
```

Arguments

`tmle_task`	a tmle_task data object
`node`	a character specifing which node to use
`likelihood`	a fitted likelihood object

discretize_variable *Discretize Continuous Variable*

Description

Converts a `data.table` column from continuous to a discrete factor

Usage

```
discretize_variable(data, variable, num_cats, breakpoints = NULL)
```

Arguments

`data`	`data.table`, containing the column to change
`variable`	`character`, the name of the column to change
`num_cats`	`integer`, the number of bins to generate
`breakpoints`	`numeric vector`, the breakpoints to use. If NULL, these will be quantiles.

Value

the updated `data.table`, modified in place

ED_from_estimates *Get Empirical Mean of EIFs from Estimates*

Description

Get Empirical Mean of EIFs from Estimates

Usage

```
ED_from_estimates(estimates)
```

Arguments

`estimates`	a list of estimates objects

LF_base *Base Class for Defining Likelihood Factors*

Description

A Likelihood factor models a conditional density function. The conditioning set is defined as all parent nodes (defined in tmle3_Task). In the case of a continuous outcome variable, where a full density isn't needed, this can also model a conditional mean. This is the base class, which is intended to be abstract. See below for a list of possible likelihood factor classes.

Format

R6Class object.

Value

LF_base object

Constructor

```
define_lf(LF_base, name, ..., type = "density")
```

name character, the name of the factor. Should match a node name in the nodes specified by tmle3_Task$npsem

... Not currently used.

type character, either "density", for conditional density or, "mean" for conditional mean

Methods

get_density(tmle_task) Get conditional density values for for the observations in tmle_task.

- tmle_task: tmle3_Task to get likelihood values for

get_mean(tmle_task) Get conditional mean values for for the observations in tmle_task.

- tmle_task: tmle3_Task to get likelihood values for

Fields

name character, the name of the factor. Should match a node name in the nodes specified by tmle3_Task$npsem

type character, either "density", for conditional density or, "mean" for conditional mean

variable_type variable_type object, specifying the data type of the outcome variable. Only available after Likelihood training.

values Possible values of the outcome variable, retrivied from the variable_type object. Only available after Likelihood training.

See Also

Other Likelihood objects: CF_Likelihood, LF_derived, LF_emp, LF_fit, LF_known, LF_static, LF_targeted, Likelihood, Targeted_Likelihood, define_lf()

LF_derived *Derived Likelihood Factor Estimated from Data + Other Likelihood values, using sl3.*

Description

Uses an sl3 learner to estimate a likelihood factor from data. Inherits from LF_base; see that page for documentation on likelihood factors in general.

Format

R6Class object.

Value

LF_base object

Constructor

```
define_lf(LF_fit, name, learner, ..., type = "density")
```

`name` character, the name of the factor. Should match a node name in the nodes specified by tmle3_Task$npsem

`learner` An sl3 learner to be used to estimate the factor

`...` Not currently used.

`type` character, either "density", for conditional density or, "mean" for conditional mean

Fields

`learner` The learner or learner fit object

See Also

Other Likelihood objects: CF_Likelihood, LF_base, LF_emp, LF_fit, LF_known, LF_static, LF_targeted, Likelihood, Targeted_Likelihood, define_lf()

LF_emp *Likelihood Factor Estimated using Empirical Distribution*

Description

Uses the empirical probability distribution (puts mass $1/n$ on each of the observations, or uses weights if specified) to estimate a marginal density. Inherits from LF_base; see that page for documentation on likelihood factors in general. Only compatible with marginal likelihoods (no parent nodes). Only compatible with densities (no conditional means). The type argument will be ignored if specified.

Format

R6Class object.

Value

LF_base object

Constructor

```
define_lf(LF_emp, name, ...)
```

name character, the name of the factor. Should match a node name in the nodes specified by tmle3_Task$npsem

... Not currently used.

See Also

Other Likelihood objects: CF_Likelihood, LF_base, LF_derived, LF_fit, LF_known, LF_static, LF_targeted, Likelihood, Targeted_Likelihood, define_lf()

LF_fit *Likelihood Factor Estimated from Data using sl3.*

Description

Uses an sl3 learner to estimate a likelihood factor from data. Inherits from LF_base; see that page for documentation on likelihood factors in general.

Format

R6Class object.

Value

LF_base object

Constructor

```
define_lf(LF_fit, name, learner, ..., type = "density")
```

`name` character, the name of the factor. Should match a node name in the nodes specified by `tmle3_Task$npsem`

`learner` An sl3 learner to be used to estimate the factor

`...` Not currently used.

`type` character, either "density", for conditional density or, "mean" for conditional mean

Fields

`learner` The learner or learner fit object

See Also

Other Likelihood objects: `CF_Likelihood`, `LF_base`, `LF_derived`, `LF_emp`, `LF_known`, `LF_static`, `LF_targeted`, `Likelihood`, `Targeted_Likelihood`, `define_lf()`

LF_known *Known True Likelihood Factor*

Description

Incorporate existing knowledge about the likelihood Inherits from `LF_base`; see that page for documentation on likelihood factors in general.

Format

`R6Class` object.

Value

`LF_base` object

Constructor

```
define_lf(LF_fit, name, mean_fun, density_fun, ..., type = "density")
```

`name` character, the name of the factor. Should match a node name in the nodes specified by `tmle3_Task$npsem`

`mean_fun` A function that takes a sl3 regression task and returns true conditional means

`density_fun` A function that takes a sl3 regression task and returns true conditional densities

`...` Not currently used.

`type` character, either "density", for conditional density or, "mean" for conditional mean

See Also

Other Likelihood objects: CF_Likelihood, LF_base, LF_derived, LF_emp, LF_fit, LF_static, LF_targeted, Likelihood, Targeted_Likelihood, define_lf()

LF_static *Static Likelihood Factor*

Description

Likelihood factor for a variable that only has one value with probability 1. This is used for static interventions. Inherits from LF_base; see that page for documentation on likelihood factors in general.

Format

R6Class object.

Value

LF_base object

Constructor

```
define_lf(LF_static, name, type, value, ...)
```

name character, the name of the factor. Should match a node name in the nodes specified by tmle3_Task$npsem

type character, either "density", for conditional density or, "mean" for conditional mean

value the static value

... Not currently used.

Fields

value the static value.

See Also

Other Likelihood objects: CF_Likelihood, LF_base, LF_derived, LF_emp, LF_fit, LF_known, LF_targeted, Likelihood, Targeted_Likelihood, define_lf()

LF_targeted *Use a likelihood factor from an existing targeted likelihood*

Description

Uses an sl3 learner to estimate a likelihood factor from data. Inherits from LF_base; see that page for documentation on likelihood factors in general.

Format

R6Class object.

Value

LF_base object

Constructor

```
define_lf(LF_fit, name, learner, ..., type = "density")
```

name character, the name of the factor. Should match a node name in the nodes specified by tmle3_Task$npsem

learner An sl3 learner to be used to estimate the factor

... Not currently used.

type character, either "density", for conditional density or, "mean" for conditional mean

Fields

learner The learner or learner fit object

See Also

Other Likelihood objects: CF_Likelihood, LF_base, LF_derived, LF_emp, LF_fit, LF_known, LF_static, Likelihood, Targeted_Likelihood, define_lf()

`Likelihood` *Class for Likelihood*

Description

This object represents an estimate of the relevant factors of the likelihood estimated from data, or based on *a priori* knowledge where appropriate. That is, it represents some subset of P_n. This object inherits from `Lrnr_base`, and so shares some properties with `sl3` learners. Specifically, to fit a likelihood object to data, one calls `likelihood$train(tmle3_task)`. Each likelihood factor is represented by an object inheriting from `LF_base`.

Usage

```
make_Likelihood(...)
```

Arguments

`...`	Passes all arguments to the constructor. See documentation for the Constructor below.

Format

`R6Class` object.

Value

`Likelihood` object

Constructor

```
make_Likelihood(factor_list, ...)
```

`factor_list` A list of objects inheriting from `LF_base`, representing the individual relevant factors.

`...` Not currently used.

Methods

`validate_task(tmle_task)` Ensure that this likelihood is compatible with a particular `tmle3_Task`, in that the factor names must match the `tmle_task$npsem` names.

- `tmle_task`: the `tmle3_Task` to validate.

`get_initial_likelihoods(tmle_task, nodes=NULL)` Gets initial (i.e. before any TMLE updates) likelihood values for the specified `nodes` (or all nodes if none are specified) for the observations in `tmle_task`.

- `tmle_task`: `tmle3_Task` to get likelihood values for
- `nodes`: character vectors, the list of nodes to get likelihood values for. If missing, values will be provided for all nodes.

`get_likelihoods(tmle_task, nodes=NULL)` Gets updated (i.e. after all TMLE updates) likelihood values for the specified `nodes` (or all nodes if none are specified) for the observations in `tmle_task`.

- `tmle_task`: `tmle3_Task` to get likelihood values for
- `nodes`: character vectors, the list of nodes to get likelihood values for. If missing, values will be provided for all nodes.

`get_possible_counterfactuals(nodes)` Gets all possible combination of counterfactual values for a set of nodes. This is useful for marginalizing over a node. Returns a `data.frame` with one row per possibility.

- `nodes`: character vectors, the list of nodes to get counterfactual values for. If missing, values will be provided for all nodes.

Fields

`factor_list` The list of `LF_base` objects specifying the relevant likelihood factors

`observed_values` The likelihood values for the observed data. These are cached, as they are used in many places in TMLE

`update_list` A list of tmle_updates that have been calculated for this likelihood

See Also

Other Likelihood objects: `CF_Likelihood`, `LF_base`, `LF_derived`, `LF_emp`, `LF_fit`, `LF_known`, `LF_static`, `LF_targeted`, `Targeted_Likelihood`, `define_lf()`

`Likelihood_cache` *Cache Likelihood values, update those values*

Description

Cache Likelihood values, update those values

`Param_ATC` *Additive Effect of Treatment Among the Treated*

Description

Parameter definition for the Additive Effect of Treatment Among the Treated (ATT). Currently supports multiple static intervention nodes. Does yet not support dynamic rule or stochastic interventions.

Format

`R6Class` object.

Value

Param_base object

Current Issues

- clever covariates doesn't support updates; always uses initial (necessary for iterative TMLE, e.g. stochastic intervention)
- doesn't integrate over possible counterfactuals (necessary for stochastic intervention)
- clever covariate gets recalculated all the time (inefficient)

Constructor

```
define_param(Param_ATT, observed_likelihood, intervention_list, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

intervention_list_treatment A list of objects inheriting from LF_base, representing the treatment intervention.

intervention_list_control A list of objects inheriting from LF_base, representing the control intervention.

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood_treatment the counterfactual likelihood for the treatment

cf_likelihood_control the counterfactual likelihood for the control

intervention_list_treatment A list of objects inheriting from LF_base, representing the treatment intervention

intervention_list_control A list of objects inheriting from LF_base, representing the control intervention

See Also

Other Parameters: Param_ATE, Param_ATT, Param_MSM, Param_TSM, Param_base, Param_delta, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

Param_ATE *Average Treatment Effect*

Description

Parameter definition for the Average Treatment Effect (ATE).

Format

R6Class object.

Value

Param_base object

Constructor

```
define_param(Param_ATT, observed_likelihood, intervention_list, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

intervention_list_treatment A list of objects inheriting from LF_base, representing the treatment intervention.

intervention_list_control A list of objects inheriting from LF_base, representing the control intervention.

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood_treatment the counterfactual likelihood for the treatment

cf_likelihood_control the counterfactual likelihood for the control

intervention_list_treatment A list of objects inheriting from LF_base, representing the treatment intervention

intervention_list_control A list of objects inheriting from LF_base, representing the control intervention

See Also

Other Parameters: Param_ATC, Param_ATT, Param_MSM, Param_TSM, Param_base, Param_delta, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

Param_ATT *Additive Effect of Treatment Among the Treated*

Description

Parameter definition for the Additive Effect of Treatment Among the Treated (ATT). Currently supports multiple static intervention nodes. Does yet not support dynamic rule or stochastic interventions.

Format

R6Class object.

Value

Param_base object

Current Issues

- clever covariates doesn't support updates; always uses initial (necessary for iterative TMLE, e.g. stochastic intervention)
- doesn't integrate over possible counterfactuals (necessary for stochastic intervention)
- clever covariate gets recalculated all the time (inefficient)

Constructor

```
define_param(Param_ATT, observed_likelihood, intervention_list, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

intervention_list_treatment A list of objects inheriting from LF_base, representing the treatment intervention.

intervention_list_control A list of objects inheriting from LF_base, representing the control intervention.

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood_treatment the counterfactual likelihood for the treatment

cf_likelihood_control the counterfactual likelihood for the control

intervention_list_treatment A list of objects inheriting from LF_base, representing the treatment intervention

intervention_list_control A list of objects inheriting from LF_base, representing the control intervention

See Also

Other Parameters: Param_ATC, Param_ATE, Param_MSM, Param_TSM, Param_base, Param_delta, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

---

Param_base *Base Class for Defining Parameters*

Description

A parameter is a function of the likelihood. Once given a Likelihood object, a parameter will a value. These objects also contain information about the efficient influence function (EIF) of a parameter, as well as its clever covariate(s).

Format

R6Class object.

Value

Param_base object

Constructor

```
define_param(Param_base, observed_likelihood, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Methods

clever_covariates(tmle_task = NULL) Get the clever covariates for an TMLE update step.

- tmle_task: tmle3_Task to get clever covariate values for. If NULL, the tmle_task used to train the observed likelihood will be used

estimates(tmle_task = NULL) Get the parameter estimates and influence curve values.

- tmle_task: tmle3_Task to get clever covariate values for. If NULL, the tmle_task used to train the observed likelihood will be used

Fields

observed_likelihood the observed likelihood

outcome_node character, the name of the outcome node

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_MSM, Param_TSM, Param_delta, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

Param_delta *Delta Method Parameters*

Description

These parameters are smooth functionals of one or more other params They are not fit directly with tmle, but are estimated using the delta method todo: better docs They do not return have clever covariates

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_MSM, Param_TSM, Param_base, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

Param_mean *Mean of Outcome Node*

Description

Parameter for marginal mean of Y: $\Psi = E[Y]$. No TMLE update needed, but can be used in delta method calculations. Useful for example, in calculating attributable risks.

Format

R6Class object.

Value

Param_base object

Constructor

```
define_param(Param_TSM, observed_likelihood, intervention_list, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood the counterfactual likelihood for this treatment

intervention_list A list of objects inheriting from LF_base, representing the intervention

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_MSM, Param_TSM, Param_base, Param_delta, Param_stratified, Param_survival, define_param(), tmle3_Fit

Param_MSM *Stratified Parameter Estimates via MSM*

Description

Stratified Parameter Estimates via MSM

Format

R6Class object.

Value

Param_base object

Current Issues

- clever covariates doesn't support updates; always uses initial (necessary for iterative TMLE, e.g. stochastic intervention)
- clever covariate gets recalculated all the time (inefficient)

Constructor

```
define_param(Param_MSM, observed_likelihood, strata_variable, ...)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

msm form of the MSM. Default is "A + V", consistent with the default of treatment_node and strata_name.

weight "Cond.Prob.", "Unif." or custom input function. Note that custom function should support vector input. Default is "Cond.Prob.".

... Not currently used.

covariate_node character, the name of the node that should be treated as the covariate

treatment_node character, the name of the node that should be treated as the treatment

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood the counterfactual likelihood for this treatment

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_TSM, Param_base, Param_delta, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

`Param_stratified` *Stratified Parameter Estimates*

Description

Stratified Parameter Estimates

Format

`R6Class` object.

Value

`Param_base` object

Current Issues

- clever covariates doesn't support updates; always uses initial (necessary for iterative TMLE, e.g. stochastic intervention)
- doesn't integrate over possible counterfactuals (necessary for stochastic intervention)
- clever covariate gets recalculated all the time (inefficient)

Constructor

```
define_param(Param_TSM, observed_likelihood, intervention_list, ..., outcome_node)
```

`observed_likelihood` A `Likelihood` corresponding to the observed likelihood

`intervention_list` A list of objects inheriting from `LF_base`, representing the intervention.

`...` Not currently used.

`outcome_node` character, the name of the node that should be treated as the outcome

Fields

`cf_likelihood` the counterfactual likelihood for this treatment

`intervention_list` A list of objects inheriting from `LF_base`, representing the intervention

See Also

Other Parameters: `Param_ATC`, `Param_ATE`, `Param_ATT`, `Param_MSM`, `Param_TSM`, `Param_base`, `Param_delta`, `Param_mean`, `Param_survival`, `define_param()`, `tmle3_Fit`

Param_survival *Survival Curve*

Description

Survival Curve

Format

R6Class object.

Value

Param_base object

Constructor

```
define_param(Param_survival, observed_likelihood, intervention_list, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

intervention_list A list of objects inheriting from LF_base, representing the intervention.

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood the counterfactual likelihood for this treatment

intervention_list A list of objects inheriting from LF_base, representing the intervention

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_MSM, Param_TSM, Param_base, Param_delta, Param_mean, Param_stratified, define_param(), tmle3_Fit

Param_TSM *Treatment Specific Mean*

Description

Parameter definition for the Treatment Specific Mean (TSM): $E_W[E_Y|A(Y|A=a|W)|$. Currently supports multiple static intervention nodes. Does yet not support dynamic rule or stochastic interventions.

Format

R6Class object.

Value

Param_base object

Current Issues

- clever covariates doesn't support updates; always uses initial (necessary for iterative TMLE, e.g. stochastic intervention)
- doesn't integrate over possible counterfactuals (necessary for stochastic intervention)
- clever covariate gets recalculated all the time (inefficient)

Constructor

```
define_param(Param_TSM, observed_likelihood, intervention_list, ..., outcome_node)
```

observed_likelihood A Likelihood corresponding to the observed likelihood

intervention_list A list of objects inheriting from LF_base, representing the intervention.

... Not currently used.

outcome_node character, the name of the node that should be treated as the outcome

Fields

cf_likelihood the counterfactual likelihood for this treatment

intervention_list A list of objects inheriting from LF_base, representing the intervention

See Also

Other Parameters: Param_ATC, Param_ATE, Param_ATT, Param_MSM, Param_base, Param_delta, Param_mean, Param_stratified, Param_survival, define_param(), tmle3_Fit

---

plot_vim *Plot results of variable importance analysis*

Description

Plot results of variable importance analysis

Usage

```
plot_vim(vim_results)
```

Arguments

vim_results	Object produced by invoking tmle3_vim.

point_tx_npsem *Helper Functions for Point Treatment*

Description

Handles the common W (covariates), A (treatment/intervention), Y (outcome) data structure

Usage

```
point_tx_npsem(node_list, variable_types = NULL)

point_tx_task(data, node_list, variable_types = NULL, ...)

point_tx_likelihood(tmle_task, learner_list)
```

Arguments

node_list	a list of character vectors, listing the variables that comprise each node
variable_types	a list of variable types, one for each node. If missing, variable types will be guessed
data	a data.frame, or data.table containing data for use in estimation
...	extra arguments.
tmle_task	a tmle3_Task as constructed via point_tx_task
learner_list	a list of sl3 learners, one for A and one for Y to be used for likelihood estimation

process_missing *Preprocess Data to Handle Missing Variables*

Description

Process data to account for missingness in preparation for TMLE

Usage

```
process_missing(
  data,
  node_list,
  complete_nodes = c("A", "Y"),
  impute_nodes = NULL,
  max_p_missing = 0.5
)
```

Arguments

data	data.table, containing the missing variables
node_list	list, what variables comprise each node
complete_nodes	character vector, nodes we must observe
impute_nodes	character vector, nodes we will impute
max_p_missing	numeric, what proportion of missing is tolerable? Beyond that, the variable will be dropped from the analysis

Details

Rows where there is missingness in any of the complete_nodes will be dropped. Then, missingness will be median-imputed for the variables in the impute_nodes. Indicator variables of missingness will be generated for these nodes.

Then covariates will be processed as follows:

1. any covariate with more than max_p_missing missingness will be dropped
2. indicators of missingness will be generated
3. missing values will be median-imputed

Value

list containing the following elements:

- data, the updated dataset
- node_list, the updated list of nodes
- n_dropped, the number of observations dropped
- dropped_cols, the variables dropped due to excessive missingness

---

submodel_logit *Logistic Submodel Fluctuation*

Description

Logistic Submodel Fluctuation

Usage

```
submodel_logit(eps, X, offset)
```

Arguments

eps	...
X	...
offset	...

summary_from_estimates *Summarize Estimates*

Description

Generates a data.table summarizing results with inference

Usage

```
summary_from_estimates(
  task,
  estimates,
  param_types = NULL,
  param_names = NULL,
  init_psi = NULL,
  simultaneous_ci = FALSE
)
```

Arguments

task	tmle3_Task containing the observed data of interest; the same as that passed to ..
estimates	list, TMLE estimates of parameter and ICs from tmle3_Fit$estimates
param_types	the types of the parameters being estimated
param_names	the names of the parameters being estimated
init_psi	the names of the parameters being estimated
simultaneous_ci	if TRUE, calculate simulatenous confidence intervals

Value

data.table summarizing results

survival_tx_npsem *Helper Functions for Survival Analysis*

Description

Handles the W (covariates), A (treatment/intervention), T_tilde (time-to-event), Delta (censoring indicator), t_max (the maximum time to estimate) survival data structure

Usage

```
survival_tx_npsem(node_list, variable_types = NULL)

survival_tx_task(data, node_list, variable_types = NULL, ...)

survival_tx_likelihood(tmle_task, learner_list)
```

Arguments

node_list	a list of character vectors, listing the variables that comprise each node
variable_types	a list of variable types, one for each node. If missing, variable types will be guessed
data	a data.frame, or data.table containing data for use in estimation
...	extra arguments.
tmle_task	a tmle3_Task as constructed via survival_tx_task
learner_list	a list of sl3 learners, one for A and one for Y to be used for likelihood estimation

---

Targeted_Likelihood *Targeted Likelihood*

Description

Represents a likelihood where one or more likelihood factors has been updated to target a set of parameter(s)

Format

R6Class object.

Value

Likelihood object

Constructor

```
make_Likelihood(factor_list, ...)
```

factor_list A list of objects inheriting from LF_base, representing the individual relevant factors.

... Not currently used.

Methods

`validate_task(tmle_task)` Ensure that this likelihood is compatible with a particular `tmle3_Task`, in that the factor names must match the `tmle_task$npsem` names.

- `tmle_task`: the `tmle3_Task` to validate.

`get_initial_likelihoods(tmle_task, nodes=NULL)` Gets initial (i.e. before any TMLE updates) likelihood values for the specified `nodes` (or all nodes if none are specified) for the observations in `tmle_task`.

- `tmle_task`: `tmle3_Task` to get likelihood values for
- `nodes`: character vectors, the list of nodes to get likelihood values for. If missing, values will be provided for all nodes.

`get_likelihoods(tmle_task, nodes=NULL)` Gets updated (i.e. after all TMLE updates) likelihood values for the specified `nodes` (or all nodes if none are specified) for the observations in `tmle_task`.

- `tmle_task`: `tmle3_Task` to get likelihood values for
- `nodes`: character vectors, the list of nodes to get likelihood values for. If missing, values will be provided for all nodes.

`get_possible_counterfactuals(nodes)` Gets all possible combination of counterfactual values for a set of nodes. This is useful for marginalizing over a node. Returns a `data.frame` with one row per possibility.

- `nodes`: character vectors, the list of nodes to get counterfactual values for. If missing, values will be provided for all nodes.

Fields

`factor_list` The list of `LF_base` objects specifying the relevant likelihood factors

`observed_values` The likelihood values for the observed data. These are cached, as they are used in many places in TMLE

`update_list` A list of tmle_updates that have been calculated for this likelihood

See Also

Other Likelihood objects: `CF_Likelihood`, `LF_base`, `LF_derived`, `LF_emp`, `LF_fit`, `LF_known`, `LF_static`, `LF_targeted`, `Likelihood`, `define_lf()`

`tmle3` *TMLE from a tmle3_Spec object*

Description

Using a tmle3_Spec object, fit a TMLE

Usage

```
tmle3(tmle_spec, data, node_list, learner_list = NULL)
```

Arguments

tmle_spec	tmle3_Spec, defines the TMLE
data	data.frame, the raw data
node_list	list, defines which variables are which nodes
learner_list	list, defines which learners are used to fit which likelihood factors

Value

A tmle3_Fit object

---

tmle3_Fit *TMLE fit object*

Description

A tmle_fit object, containing initial and updated estimates, as well as data about the fitting procedure. TMLE updates are calculated when the object is constructed.

Usage

```
fit_tmle3(...)
```

Arguments

...	Passes all arguments to the constructor. See documentation for the Constructor.

Format

R6Class object.

Value

Param_base object

Constructor

```
fit_tmle3(tmle_task, likelihood, tmle_params, updater, max_it=100, ...)
```

tmle_task A tmle3_Task object defining the data and NP-SEM

likelihood A Likelihood object defining the factorized likelihood

tmle_params A list of parameters inheriting from Param_base defining the parameter(s) of interest

updater A tmle3_Update object defining the update procedure, including submodel and loss function

maxit integer, maximum number of TMLE iterations

... Not currently used.

Methods

`set_timings(start_time, task_time, likelihood_time, params_time, fit_time)` Provide the timings for the different steps of the TMLE procedure, for later reporting to the user

- `tmle_task`: `tmle3_Task` to get clever covariate values for. If NULL, the `tmle_task` used to train the `observed likelihood will be used`

`estimates(tmle_task = NULL)` Get the parameter estimates and influence curve values.

- `tmle_task`: `tmle3_Task` to get clever covariate values for. If NULL, the `tmle_task` used to train the `observed likelihood will be used`

Fields

`tmle_task` A `tmle3_Task` object defining the data and NP-SEM

`likelihood` A `Likelihood` object defining the factorized likelihood

`tmle_params` A list of parameters inheriting from `Param_base` defining the parameter(s) of interest

`tmle_names` A list of parameter names, obtained by calling `param$name` on each parameter

`updater` A `tmle3_Update` object defining the update procedure, including submodel and loss function

`steps` integer, he number of steps until TMLE converged

`ED` vector, the mean of the EIF for all the parameters

`initial_psi` vector, the initial parameter estimates

`estimates` list, final parameter estimates and ICs

`summary` data.table, summary of results

`timings` data.frame, timings for each step (provided by `tmle3_Fit$set_timings`)

See Also

Other Parameters: `Param_ATC`, `Param_ATE`, `Param_ATT`, `Param_MSM`, `Param_TSM`, `Param_base`, `Param_delta`, `Param_mean`, `Param_stratified`, `Param_survival`, `define_param()`

---

tmle3_Node *A Node (set of variables) in an NPSEM*

Description

This class defines a node in an NPSEM

Usage

```
define_node(...)
```

Arguments

...	Passes all arguments to the constructor. See documentation for the Constructor below.

Format

R6Class object.

Value

tmle3_Node object

Constructor

```
make_tmle3_task(name, variables, parents = c(), variable_type = NULL)
```

name character, the name of node

variables character vector, the names of the variables that comprise the node

parents character vector, the names of the parent nodes. If censoring, node is assumed to have no parents.

variable_type variable_type object, specifying the data type of this variable. If censoring, variable_type will be guessed later from the data.

Methods

guess_variable_type(variable_data) Guesses the variable_type from the provided data. This will be called by the tmle3_Task constructor if no variable_type was provided.

- variable_data: the observed variable data.

Fields

name character, the name of node

variables character vector, the names of the variables that comprise the node

parents character vector, the names of the parent nodes. If censoring, node is assumed to have no parents.

variable_type variable_type object, specifying the data type of this variable.

---

tmle3_Spec *Defines a TML Estimator (except for the data)*

Description

Current limitations: pretty much tailored to Param_TSM

---

tmle3_Spec_ATC *Defines a TML Estimator (except for the data)*

Description

Defines a TML Estimator (except for the data)

tmle3_Spec_ATE *Defines a TML Estimator (except for the data)*

Description

Defines a TML Estimator (except for the data)

tmle3_Spec_ATT *Defines a TML Estimator (except for the data)*

Description

Defines a TML Estimator (except for the data)

tmle3_Spec_MSM *Defines a Stratified TML Estimator with MSM (except for the data)*

Description

Defines a Stratified TML Estimator with MSM (except for the data)

tmle3_Spec_OR *Defines a TML Estimator for the Odds Ratio*

Description

Current limitations: pretty much tailored to Param_TSM see TODOs for places generalization can be added

tmle3_Spec_PAR *Defines a tmle (minus the data)*

Description

Current limitations: pretty much tailored to Param_TSM see TODOs for places generalization can be added

`tmle3_Spec_RR` *Defines a TML Estimator for the Risk Ratio*

Description

Current limitations: pretty much tailored to Param_TSM see TODOs for places generalization can be added

`tmle3_Spec_stratified` *Defines a Stratified TML Estimator (except for the data)*

Description

Defines a Stratified TML Estimator (except for the data)

`tmle3_Spec_survival` *Defines a TML Estimator (except for the data)*

Description

Defines a TML Estimator (except for the data)

`tmle3_Spec_TSM_all` *Defines a TML Estimator (except for the data)*

Description

Current limitations: pretty much tailored to `Param_TSM` See TODOs for places generalization can be added

`tmle3_Task` *Class for Storing Data and NPSEM for TMLE*

Description

This class inherits from `sl3_Task`. In addition to all the methods supported by `sl3_Task`, it supports the following.

Usage

```
make_tmle3_Task(...)
```

Arguments

`...`	Passes all arguments to the constructor. See documentation for the Constructor below.

Format

`R6Class` object.

Value

`tmle3_Task` object

Constructor

```
make_tmle3_task(data, npsem, ...)
```

`data` A `data.frame` or `data.table` containing the underlying data

`npsem` A list of `tmle3_Node` objects, where each is created using `define_node`. These specify the NPSEM. See examples.

`...` Other arguments passed to the constructor of `sl3_Task`. **NB:** Support for these is currently limited.

Methods

`get_tmle_node(node_name, bound = FALSE)` Gets the data associated with a tmle_node. Bounds the data if requested.

- `node_name`: character, the name of the node to get.
- `bound`: logical, if true the data is transformed to be in (0,1) based on pre-specified bounds.

`get_regression_task(target_node, bound = FALSE)` Gets a `sl3_Task` suitable for fitting the conditional likelihood factor with the `target_node` as the outcome.

- `target_node`: character, the name of the node to get.

`generate_counterfacutal_task(uuid, new_data)` Generates a new `tmle_Task` where some nodes are overridden to have counterfactual values.

- uuid: A unique identifier for the counterfactual task, as generated by UUIDgenerate
new_data: A data.frame or data.table with the counterfactual values. Column names must refer to node names in the npsem for this task.

Fields

npsem The list of tmle3_Node objects specifying the NPSEM

---

tmle3_Update *Defines an update procedure (submodel+loss function)*

Description

Current Limitations: loss function and submodel are hard-coded (need to accept arguments for these)

Constructor

```
define_param(maxit, cvtmle, one_dimensional, constrain_step, delta_epsilon, verbose)
```

maxit The maximum number of update iterations

cvtmle If TRUE, use CV-likelihood values when calculating updates.

one_dimensional If TRUE, collapse clever covariates into a one-dimensional clever covariate scaled by the mean of their EIFs.

constrain_step If TRUE, step size is at most delta_epsilon (it can be smaller if a smaller step decreases the loss more).

delta_epsilon The maximum step size allowed if constrain_step is TRUE.

convergence_type The convergence criterion to use: (1) "scaled_var" corresponds to sqrt(Var(D)/n)/logn (the default) while (2) "sample_size" corresponds to 1/n.

fluctuation_type Whether to include the auxiliary covariate for the fluctuation model as a covariate or to treat it as a weight. Note that the option "weighted" is incompatible with a multi-epsilon submodel (one_dimensional = FALSE).

use_best If TRUE, the final updated likelihood is set to the likelihood that minimizes the ED instead of the likelihood at the last update step.

verbose If TRUE, diagnostic output is generated about the updating procedure.

tmle3_Update_survival *Defines an update procedure (submodel+loss function) for survival data*

Description

Current Limitations: loss function and submodel are hard-coded (need to accept arguments for these)

Constructor

`define_param(maxit, cvtmle, one_dimensional, constrain_step, delta_epsilon, verbose)`

`maxit` The maximum number of update iterations

`cvtmle` If `TRUE`, use CV-likelihood values when calculating updates.

`one_dimensional` If `TRUE`, collapse clever covariates into a one-dimensional clever covariate scaled by the mean of their EIFs.

`constrain_step` If `TRUE`, step size is at most `delta_epsilon` (it can be smaller if a smaller step decreases the loss more).

`delta_epsilon` The maximum step size allowed if `constrain_step` is `TRUE`.

`convergence_type` The convergence criterion to use: (1) `"scaled_var"` corresponds to sqrt(Var(D)/n)/logn (the default) while (2) `"sample_size"` corresponds to 1/n.

`fluctuation_type` Whether to include the auxiliary covariate for the fluctuation model as a covariate or to treat it as a weight. Note that the option `"weighted"` is incompatible with a multi-epsilon submodel (`one_dimensional = FALSE`).

`verbose` If `TRUE`, diagnostic output is generated about the updating procedure.

tmle3_vim *Compute Variable Importance Measures (VIM) with any given parameter*

Description

Compute Variable Importance Measures (VIM) with any given parameter

Usage

```
tmle3_vim(
  tmle_spec,
  data,
  node_list,
  learner_list = NULL,
  adjust_for_other_A = TRUE
)
```

Arguments

tmle_spec	tmle3_Spec, defines the TMLE
data	data.frame, the raw data
node_list	list, defines which variables are which nodes
learner_list	list, defines which learners are used to fit which likelihood factors
adjust_for_other_A	Whether or not to adjust for other specified intervention nodes.

---

tmle_ATC *All Treatment Specific Means*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_ATC(treatment_level, control_level)
```

Arguments

treatment_level	the level of A that corresponds to treatment
control_level	the level of A that corresponds to a control or reference level

---

tmle_ATE *All Treatment Specific Means*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_ATE(treatment_level, control_level)
```

Arguments

treatment_level	the level of A that corresponds to treatment
control_level	the level of A that corresponds to a control or reference level

tmle_ATT *All Treatment Specific Means*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_ATT(treatment_level, control_level)
```

Arguments

treatment_level	the level of A that corresponds to treatment
control_level	the level of A that corresponds to a control or reference level

tmle_MSM *Make MSM version of Stratified TML estimator class*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_MSM(weight = "Cond.Prob.", n_samples = 30)
```

Arguments

weight	h(A, V)
n_samples	number of samples to draw for each observation if A is continuous

tmle_OR *Odds Ratio*

Description

O = (W, A, Y) W = Covariates A = Treatment (binary or categorical) Y = Outcome (binary or bounded continuous)

Usage

```
tmle_OR(baseline_level, contrast_level)
```

Arguments

baseline_level	The baseline risk group.
contrast_level	The contrast risk group.

tmle_PAR *PAR and PAF*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_PAR(baseline_level)
```

Arguments

baseline_level	the baseline risk group

tmle_RR *Risk Ratio*

Description

O = (W, A, Y) W = Covariates A = Treatment (binary or categorical) Y = Outcome (binary or bounded continuous)

Usage

```
tmle_RR(baseline_level, contrast_level)
```

Arguments

baseline_level	The baseline risk group.
contrast_level	The contrast risk group.

tmle_stratified *Stratified version of TML estimator from other Spec classes*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_stratified(base_spec, base_estimate = TRUE)
```

Arguments

base_spec	An underlying spec to stratify.
base_estimate	Indicate whether to report base parameter.

tmle_survival *Treatment Specific Survival*

Description

See the associated handbook chapter

Usage

```
tmle_survival(treatment_level, control_level, target_times = NULL, ...)
```

Arguments

treatment_level	the level of A that corresponds to treatment
control_level	the level of A that corresponds to a control or reference level
target_times	the time points to be targeted at during the TMLE adjustment
...	others args passed to spec

tmle_TSM_all *All Treatment Specific Means*

Description

O=(W,A,Y) W=Covariates A=Treatment (binary or categorical) Y=Outcome (binary or bounded continuous)

Usage

```
tmle_TSM_all()
```

train_lf *Manually Train Likelihood Factor The internal training process for likelihood factors is somewhat obtuse, so this function does the steps to manually train one, which is helpful if you want to use a likelihood factor independently of a likelihood object*

Description

Manually Train Likelihood Factor The internal training process for likelihood factors is somewhat obtuse, so this function does the steps to manually train one, which is helpful if you want to use a likelihood factor independently of a likelihood object

Usage

```
train_lf(lf, tmle_task)
```

Arguments

`lf`	the likelihood factor to train
`tmle_task`	the task to use for training

Index