Package 'TrialEmulation'

November 19, 2025

Title Causal Analysis of Observational Time-to-Event Data

Version 0.0.4.8

Description Implements target trial emulation methods to apply randomized clinical trial design and analysis in an observational setting. Using marginal structural models, it can estimate intention-to-treat and per-protocol effects in emulated trials using electronic health records. A description and application of the method can be found in Danaei et al (2013) <doi:10.1177/0962280211403603>.

License Apache License (>= 2)

URL https://causal-lda.github.io/TrialEmulation/, https://github.com/Causal-LDA/TrialEmulation

BugReports https://github.com/Causal-LDA/TrialEmulation/issues

Depends R (>= 4.1.0)

Imports broom (>= 0.7.10), checkmate, data.table (>= 1.9.8), DBI, duckdb, formula.tools, lifecycle, lmtest, methods, mvtnorm, parglm, Rcpp, sandwich

Suggests knitr, parsnip, rmarkdown, rpart, testthat (>= 3.0.0), withr

LinkingTo Rcpp

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

Collate 'RcppExports.R' 'calculate_weights.R' 'data.R' 'data_extension.R' 'data_manipulation.R' 'data_preparation.R' 'data_simulation.R' 'data_utils.R' 'expand_trials.R' 'generics.R' 'initiators.R' 'lr_utils.R' 'te_model_fitter.R' 'te_outcome_model.R' 'te_datastore.R' 'te_weights.R'

'te_data.R' 'te_expansion.R' 'trial_sequence.R' 'modelling.R'
'package.R' 'predict.R' 'robust.R' 'sampling.R'
'te_datastore_csv.R' 'te_datastore_duckdb.R' 'te_parsnip.R'
'te_stats_glm_logit.R' 'utils.R' 'weighting.R'

NeedsCompilation yes

Author Isaac Gravestock [aut, cre] (ORCID:
<https://orcid.org/0000-0003-0283-2065>),
Li Su [aut],
Roonak Rezvani [aut] (ORCID: <https://orcid.org/0000-0001-5580-5058>,
Original package author),
Julia Moesch [aut],
Medical Research Council (MRC) [fnd],
F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Isaac Gravestock <isaac.gravestock@roche.com>

Repository CRAN

Date/Publication 2025-11-19 08:50:02 UTC

Contents

`calculate_weights` *Calculate Inverse Probability of Censoring Weights*

Description

[Experimental]

Usage

```
calculate_weights(object, ...)

## S4 method for signature 'trial_sequence_ITT'
calculate_weights(object, quiet = FALSE)

## S4 method for signature 'trial_sequence_AT'
calculate_weights(object, quiet = FALSE)

## S4 method for signature 'trial_sequence_PP'
calculate_weights(object, quiet = FALSE)
```

Arguments

`object`	A trial_sequence object
`...`	Other arguments used by methods.
`quiet`	Prints model summaries is `TRUE`.

Value

A trial_sequence object with updated `censor_weights` and/or `switch_weights` slots

Examples

```
save_dir <- file.path(tempdir(), "switch_models")
ts <- trial_sequence("PP") |>
  set_data(
    data = data_censored,
    id = "id",
    period = "period",
    treatment = "treatment",
    outcome = "outcome",
    eligible = "eligible"
  ) |>
  set_switch_weight_model(
    numerator = ~ age + x1 + x3,
    denominator = ~age,
    model_fitter = stats_glm_logit(save_path = save_dir)
  ) |>
  calculate_weights()
```

`case_control_sampling_trials` *Case-control sampling of expanded data for the sequence of emulated trials*

Description

[Stable]

Usage

```
case_control_sampling_trials(
  data_prep,
  p_control = NULL,
  subset_condition,
  sort = FALSE
)
```

Arguments

`data_prep`	Result from `data_preparation()`.
`p_control`	Control sampling probability for selecting potential controls at each follow-up time of each trial.
`subset_condition`	Expression used to `subset()` the trial data before case-control sampling.
`sort`	Sort data before applying case-control sampling to make sure that the resulting data are identical when sampling from the expanded data created with `separate_files = TRUE` or `separate_files = FALSE`.

Details

Perform case-control sampling of expanded data to create a data set of reduced size and calculate sampling weights to be used in `trial_msm()`.

Value

A `data.frame` or a `split()` `data.frame` if `length(p_control) > 1`. An additional column `sample_weight` containing the sample weights will be added to the result. These can be included in the models fit with `trial_msm()`.

Examples

```
# If necessary reduce the number of threads for data.table
data.table::setDTthreads(2)

data("te_data_ex")
samples <- case_control_sampling_trials(te_data_ex, p_control = 0.01)
```

---

data_censored *Example of longitudinal data for sequential trial emulation containing censoring*

Description

This data contains data from 89 patients followed for up to 19 periods.

Usage

```
data_censored
```

Format

A data frame with 725 rows and 12 variables:

id patient identifier

period time period

treatment indicator for receiving treatment in this period, 1=treatment, 0=non-treatment

x1 A time-varying categorical variable relating to treatment and the outcome

x2 A time-varying numeric variable relating to treatment and the outcome

x3 A fixed categorical variable relating to treatment and the outcome

x4 A fixed categorical variable relating to treatment and the outcome

age patient age in years

age_s patient age

outcome indicator for outcome in this period, 1=event occurred, 0=no event

censored indicator for patient being censored in this period, 1=censored, 0=not censored

eligible indicator for eligibility for trial start in this period, 1=yes, 0=no

`data_preparation` *Prepare data for the sequence of emulated target trials*

Description

[Stable]

Usage

```
data_preparation(
  data,
  id = "id",
  period = "period",
  treatment = "treatment",
  outcome = "outcome",
  eligible = "eligible",
  model_var = NULL,
  outcome_cov = ~1,
  estimand_type = c("ITT", "PP", "As-Treated"),
  switch_n_cov = ~1,
  switch_d_cov = ~1,
  first_period = NA,
  last_period = NA,
  use_censor_weights = FALSE,
  cense = NA,
  pool_cense = c("none", "both", "numerator"),
  cense_d_cov = ~1,
  cense_n_cov = ~1,
  eligible_wts_0 = NA,
  eligible_wts_1 = NA,
  where_var = NULL,
  data_dir,
  save_weight_models = FALSE,
  glm_function = "glm",
  chunk_size = 500,
  separate_files = FALSE,
  quiet = FALSE,
  ...
)
```

Arguments

`data`	A `data.frame` containing all the required variables in the person-time format, i.e., the 'long' format.
`id`	Name of the variable for identifiers of the individuals. Default is 'id'.
`period`	Name of the variable for the visit/period. Default is 'period'.

`treatment` Name of the variable for the treatment indicator at that visit/period. Default is 'treatment'.

`outcome` Name of the variable for the indicator of the outcome event at that visit/period. Default is 'outcome'.

`eligible` Name of the variable for the indicator of eligibility for the target trial at that visit/period. Default is 'eligible'.

`model_var` Treatment variables to be included in the marginal structural model for the emulated trials. `model_var = "assigned_treatment"` will create a variable `assigned_treatment` that is the assigned treatment at the trial baseline, typically used for ITT and per-protocol analyses. `model_var = "dose"` will create a variable `dose` that is the cumulative number of treatments received since the trial baseline, typically used in as-treated analyses.

`outcome_cov` A RHS formula with baseline covariates to be adjusted for in the marginal structural model for the emulated trials. Note that if a time-varying covariate is specified in `outcome_cov`, only its value at each of the trial baselines will be included in the expanded data.

`estimand_type` Specify the estimand for the causal analyses in the sequence of emulated trials. `estimand_type = "ITT"` will perform intention-to-treat analyses, where treatment switching after trial baselines are ignored. `estimand_type = "PP"` will perform per-protocol analyses, where individuals' follow-ups are artificially censored and inverse probability of treatment weighting is applied. `estimand_type = "As-Treated"` will fit a standard marginal structural model for all possible treatment sequences, where individuals' follow-ups are not artificially censored but treatment switching after trial baselines are accounted for by applying inverse probability of treatment weighting.

`switch_n_cov` A RHS formula to specify the logistic models for estimating the numerator terms of the inverse probability of treatment weights. A derived variable named `time_on_regime` containing the duration of time that the individual has been on the current treatment/non-treatment is available for use in these models.

`switch_d_cov` A RHS formula to specify the logistic models for estimating the denominator terms of the inverse probability of treatment weights.

`first_period` First time period to be set as trial baseline to start expanding the data.

`last_period` Last time period to be set as trial baseline to start expanding the data.

`use_censor_weights` Require the inverse probability of censoring weights. If `use_censor_weights = TRUE`, then the variable name of the censoring indicator needs to be provided in the argument `cense`.

`cense` Variable name for the censoring indicator. Required if `use_censor_weights = TRUE`.

`pool_cense` Fit pooled or separate censoring models for those treated and those untreated at the immediately previous visit. Pooling can be specified for the models for the numerator and denominator terms of the inverse probability of censoring weights. One of `"none"`, `"numerator"`, or `"both"` (default is `"none"` except when `estimand_type = "ITT"` then default is `"numerator"`).

cense_d_cov — A RHS formula to specify the logistic models for estimating the denominator terms of the inverse probability of censoring weights.

cense_n_cov — A RHS formula to specify the logistic models for estimating the numerator terms of the inverse probability of censoring weights.

eligible_wts_0 — See definition for eligible_wts_1

eligible_wts_1 — Exclude some observations when fitting the models for the inverse probability of treatment weights. For example, if it is assumed that an individual will stay on treatment for at least 2 visits, the first 2 visits after treatment initiation by definition have a probability of staying on the treatment of 1.0 and should thus be excluded from the weight models for those who are on treatment at the immediately previous visit. Users can define a variable that indicates that these 2 observations are ineligible for the weight model for those who are on treatment at the immediately previous visit and add the variable name in the argument eligible_wts_1. Similar definitions are applied to eligible_wts_0 for excluding observations when fitting the models for the inverse probability of treatment weights for those who are not on treatment at the immediately previous visit.

where_var — Specify the variable names that will be used to define subgroup conditions when fitting the marginal structural model for a subgroup of individuals. Need to specify jointly with the argument where_case.

data_dir — Directory to save model objects when save_weight_models=TRUE and expanded data as separate CSV files names as trial_i.csvs if separate_files = TRUE. If the specified directory does not exist it will be created. If the directory already contains trial files, an error will occur, other files may be overwritten.

save_weight_models — Save model objects for estimating the weights in data_dir.

glm_function — Specify which glm function to use for the marginal structural model from the stats or parglm packages. The default function is the glm function in the stats package. Users can also specify glm_function = "parglm" such that the parglm function in the parglm package can be used for fitting generalized linear models in parallel. The default control setting for parglm is nthreads = 4 and method = "FAST", where four cores and Fisher information are used for faster computation. Users can change the default control setting by passing the arguments nthreads and method in the parglm.control function of the parglm package, or alternatively, by passing a control argument with a list produced by parglm.control(nthreads = , method =).

chunk_size — Number of individuals whose data to be processed in one chunk when separate_files = TRUE

separate_files — Save expanded data in separate CSV files for each trial.

quiet — Suppress the printing of progress messages and summaries of the fitted models.

... — Additional arguments passed to glm_function. This may be used to specify initial values of parameters or arguments to control. See stats::glm, parglm::parglm and parglm::parglm.control() for more information.

Details

This function expands observational data in the person-time format (i.e., the 'long' format) to emulate a sequence of target trials and also estimates the inverse probability of treatment and censoring weights as required.

The arguments `chunk_size` and `separate_files` allow for processing of large datasets that would not fit in memory once expanded. When `separate_files = TRUE`, the input data are processed in chunks of individuals and saved into separate files for each emulated trial. These separate files can be sampled by case-control sampling to create a reduced dataset for the modelling.

Value

An object of class `TE_data_prep`, which can either be sampled from (case_control_sampling_trials) or directly used in a model (trial_msm). It contains the elements

data the expanded dataset for all emulated trials. If `separate_files = FALSE`, it is a `data.table`; if `separate_files = TRUE`, it is a character vector with the file path of the expanded data as CSV files.

min_period index for the first trial in the expanded data

max_period index for the last trial in the expanded data

N the total number of observations in the expanded data

data_template a zero-row `data.frame` with the columns and attributes of the expanded data

switch_models a list of summaries of the models fitted for inverse probability of treatment weights, if `estimand_type` is `"PP"` or `"As-Treated"`

censor_models a list of summaries of the models fitted for inverse probability of censoring weights, if `use_censor_weights=TRUE`

args a list contain the parameters used to prepare the data and fit the weight models

---

`expand_trials` *Expand trials*

Description

[Experimental]

Usage

```
expand_trials(object)
```

Arguments

`object`	A trial_sequence object

Value

The trial_sequence `object` with a data set containing the full sequence of target trials. The data is stored according to the options set with `set_expansion_options()` and especially the `save_to_*` function.

`fit_msm` *Fit the marginal structural model for the sequence of emulated trials*

Description

[Experimental]

Usage

```
fit_msm(
  object,
  weight_cols = c("weight", "sample_weight"),
  modify_weights = NULL
)

## S4 method for signature 'trial_sequence'
fit_msm(
  object,
  weight_cols = c("weight", "sample_weight"),
  modify_weights = NULL
)
```

Arguments

`object`	A `trial_sequence` object
`weight_cols`	character vector of column names in expanded outcome dataset, ie `outcome_data(object)`. If multiple columns are specified, the element wise product will be used. Specify `NULL` if no weight columns should be used.
`modify_weights`	a function to transform the weights (or `NULL` for no transformation). Must take a numeric vector of weights and a vector of positive, finite weights of the same length. See examples for some possible function definitions. Before the outcome marginal structural model can be fit, the outcome model must be specified with `set_outcome_model()` and the data must be expanded into the trial sequence with `expand_trials()`. The model is fit based on the `model_fitter` specified in set_outcome_model using the internal `fit_outcome_model` method.

Value

A modified `trial_sequence` object with updated `outcome_model` slot.

Examples

```
trial_seq_object <- trial_sequence("ITT") |>
  set_data(data_censored) |>
  set_outcome_model(
    adjustment_terms = ~age_s,
```

```
    followup_time_terms = ~ stats::poly(followup_time, degree = 2)
  ) |>
  set_expansion_options(output = save_to_datatable(), chunk_size = 500) |>
  expand_trials() |>
  load_expanded_data()

fit_msm(trial_seq_object)

# Using modify_weights functions ----

# returns a function that truncates weights to limits
limit_weight <- function(lower_limit, upper_limit) {
  function(w) {
    w[w > upper_limit] <- upper_limit
    w[w < lower_limit] <- lower_limit
    w
  }
}

# calculate 1st and 99th percentile limits and truncate
p99_weight <- function(w) {
  p99 <- quantile(w, prob = c(0.01, 0.99), type = 1)
  limit_weight(p99[1], p99[2])(w)
}

# set all weights to 1
all_ones <- function(w) {
  rep(1, length(w))
}

fit_msm(trial_seq_object, modify_weights = limit_weight(0.01, 4))
fit_msm(trial_seq_object, modify_weights = p99_weight)
```

`fit_weights_model` *Method for fitting weight models*

Description

Method for fitting weight models

Usage

```
fit_weights_model(object, data, formula, label)
```

Arguments

`object`	The object determining which method should be used, containing any slots containing user defined parameters.
`data`	`data.frame` containing outcomes and covariates as defined in `formula`.
`formula`	`formula` describing the model.
`label`	A short string describing the model.

Value

An object of class te_weights_fitted

Examples

```
fitter <- stats_glm_logit(tempdir())
data(data_censored)
# Not usually called directly by a user
fitted <- fit_weights_model(
  object = fitter,
  data = data_censored,
  formula = 1 - censored ~ x1 + age_s + treatment,
  label = "Example model for censoring"
)
fitted
unlink(fitted@summary$save_path$path)
```

---

initiators *A wrapper function to perform data preparation and model fitting in a sequence of emulated target trials*

Description

[Stable]

Usage

```
initiators(
  data,
  id = "id",
  period = "period",
  treatment = "treatment",
  outcome = "outcome",
  eligible = "eligible",
  outcome_cov = ~1,
  estimand_type = c("ITT", "PP", "As-Treated"),
  model_var = NULL,
  switch_n_cov = ~1,
  switch_d_cov = ~1,
  first_period = NA,
  last_period = NA,
  first_followup = NA,
  last_followup = NA,
  use_censor_weights = FALSE,
  save_weight_models = FALSE,
  analysis_weights = c("asis", "unweighted", "p99", "weight_limits"),
  weight_limits = c(0, Inf),
  cense = NA,
```

```
  pool_cense = c("none", "both", "numerator"),
  cense_d_cov = ~1,
  cense_n_cov = ~1,
  include_followup_time = ~followup_time + I(followup_time^2),
  include_trial_period = ~trial_period + I(trial_period^2),
  eligible_wts_0 = NA,
  eligible_wts_1 = NA,
  where_var = NULL,
  where_case = NA,
  data_dir,
  glm_function = "glm",
  quiet = FALSE,
  ...
)
```

Arguments

data	A data.frame containing all the required variables in the person-time format, i.e., the 'long' format.
id	Name of the variable for identifiers of the individuals. Default is 'id'.
period	Name of the variable for the visit/period. Default is 'period'.
treatment	Name of the variable for the treatment indicator at that visit/period. Default is 'treatment'.
outcome	Name of the variable for the indicator of the outcome event at that visit/period. Default is 'outcome'.
eligible	Name of the variable for the indicator of eligibility for the target trial at that visit/period. Default is 'eligible'.
outcome_cov	A RHS formula with baseline covariates to be adjusted for in the marginal structural model for the emulated trials. Note that if a time-varying covariate is specified in outcome_cov, only its value at each of the trial baselines will be included in the expanded data.
estimand_type	Specify the estimand for the causal analyses in the sequence of emulated trials. estimand_type = "ITT" will perform intention-to-treat analyses, where treatment switching after trial baselines are ignored. estimand_type = "PP" will perform per-protocol analyses, where individuals' follow-ups are artificially censored and inverse probability of treatment weighting is applied. estimand_type = "As-Treated" will fit a standard marginal structural model for all possible treatment sequences, where individuals' follow-ups are not artificially censored but treatment switching after trial baselines are accounted for by applying inverse probability of treatment weighting.
model_var	Treatment variables to be included in the marginal structural model for the emulated trials. model_var = "assigned_treatment" will create a variable assigned_treatment that is the assigned treatment at the trial baseline, typically used for ITT and per-protocol analyses. model_var = "dose" will create a variable dose that is the cumulative number of treatments received since the trial baseline, typically used in as-treated analyses.

switch_n_cov A RHS formula to specify the logistic models for estimating the numerator terms of the inverse probability of treatment weights. A derived variable named time_on_regime containing the duration of time that the individual has been on the current treatment/non-treatment is available for use in these models.

switch_d_cov A RHS formula to specify the logistic models for estimating the denominator terms of the inverse probability of treatment weights.

first_period First time period to be set as trial baseline to start expanding the data.

last_period Last time period to be set as trial baseline to start expanding the data.

first_followup First follow-up time/visit in the trials to be included in the marginal structural model for the outcome event.

last_followup Last follow-up time/visit in the trials to be included in the marginal structural model for the outcome event.

use_censor_weights
Require the inverse probability of censoring weights. If use_censor_weights = TRUE, then the variable name of the censoring indicator needs to be provided in the argument cense.

save_weight_models
Save model objects for estimating the weights in data_dir.

analysis_weights
Choose which type of weights to be used for fitting the marginal structural model for the outcome event.

- "asis": use the weights as calculated.
- "p99": use weights truncated at the 1st and 99th percentiles (based on the distribution of weights in the entire sample).
- "weight_limits": use weights truncated at the values specified in weight_limits.
- "unweighted": set all analysis weights to 1, even if treatment weights or censoring weights were calculated.

weight_limits Lower and upper limits to truncate weights, given as c(lower, upper)

cense Variable name for the censoring indicator. Required if use_censor_weights = TRUE.

pool_cense Fit pooled or separate censoring models for those treated and those untreated at the immediately previous visit. Pooling can be specified for the models for the numerator and denominator terms of the inverse probability of censoring weights. One of "none", "numerator", or "both" (default is "none" except when estimand_type = "ITT" then default is "numerator").

cense_d_cov A RHS formula to specify the logistic models for estimating the denominator terms of the inverse probability of censoring weights.

cense_n_cov A RHS formula to specify the logistic models for estimating the numerator terms of the inverse probability of censoring weights.

include_followup_time
The model to include the follow up time/visit of the trial (followup_time) in the marginal structural model, specified as a RHS formula.

include_trial_period
The model to include the trial period (trial_period) in the marginal structural model, specified as a RHS formula.

`eligible_wts_0` See definition for `eligible_wts_1`

`eligible_wts_1` Exclude some observations when fitting the models for the inverse probability of treatment weights. For example, if it is assumed that an individual will stay on treatment for at least 2 visits, the first 2 visits after treatment initiation by definition have a probability of staying on the treatment of 1.0 and should thus be excluded from the weight models for those who are on treatment at the immediately previous visit. Users can define a variable that indicates that these 2 observations are ineligible for the weight model for those who are on treatment at the immediately previous visit and add the variable name in the argument `eligible_wts_1`. Similar definitions are applied to `eligible_wts_0` for excluding observations when fitting the models for the inverse probability of treatment weights for those who are not on treatment at the immediately previous visit.

`where_var` Specify the variable names that will be used to define subgroup conditions when fitting the marginal structural model for a subgroup of individuals. Need to specify jointly with the argument `where_case`.

`where_case` Define conditions using variables specified in `where_var` when fitting a marginal structural model for a subgroup of the individuals. For example, if `where_var= "age"`, `where_case = "age >= 30"` will only fit the marginal structural model to the subgroup of individuals. who are 30 years old or above.

`data_dir` Directory to save model objects in.

`glm_function` Specify which glm function to use for the marginal structural model from the `stats` or `parglm` packages. The default function is the `glm` function in the `stats` package. Users can also specify `glm_function = "parglm"` such that the `parglm` function in the `parglm` package can be used for fitting generalized linear models in parallel. The default control setting for `parglm` is `nthreads = 4` and `method = "FAST"`, where four cores and Fisher information are used for faster computation. Users can change the default control setting by passing the arguments `nthreads` and `method` in the `parglm.control` function of the `parglm` package, or alternatively, by passing a `control` argument with a list produced by `parglm.control(nthreads = , method = )`.

`quiet` Suppress the printing of progress messages and summaries of the fitted models.

`...` Additional arguments passed to `glm_function`. This may be used to specify initial values of parameters or arguments to `control`. See stats::glm, parglm::parglm and `parglm::parglm.control()` for more information.

Details

An all-in-one analysis using a sequence of emulated target trials. This provides a simplified interface to the main functions `data_preparation()` and `trial_msm()`.

Value

Returns the result of `trial_msm()` from the expanded data. An object of class `TE_msm` containing

model a `glm` object

robust a list containing a summary table of estimated regression coefficients and the robust covariance matrix

ipw_data *IPW Data Accessor and Setter*

Description

[Experimental]

Usage

```
ipw_data(object)

ipw_data(object) <- value

## S4 method for signature 'trial_sequence'
ipw_data(object)

## S4 replacement method for signature 'trial_sequence'
ipw_data(object) <- value
```

Arguments

object	trial_sequence object
value	data.table to replace and update in @data

Details

Generic function to access and update the data used for inverse probability weighting.

The setter method ipw_data(object) <- value does not perform the same checks and manipulations as set_data(). To completely replace the data please use set_data(). This ipw_data<- method allows small changes such as adding a new column.

Value

The data from the @data slot of object used for inverse probability weighting.

Examples

```
ts <- trial_sequence("ITT")
ts <- set_data(ts, data_censored)
ipw_data(ts)
data.table::set(ipw_data(ts), j = "dummy", value = TRUE)

# or with the setter method:
new_data <- ipw_data(ts)
new_data$x2sq <- new_data$x2^2
ipw_data(ts) <- new_data
```

load_expanded_data *Method to read, subset and sample expanded data*

Description

[Experimental]

Usage

```
load_expanded_data(
  object,
  p_control = NULL,
  period = NULL,
  subset_condition = NULL,
  seed = NULL
)

## S4 method for signature 'trial_sequence'
load_expanded_data(
  object,
  p_control = NULL,
  period = NULL,
  subset_condition = NULL,
  seed = NULL
)
```

Arguments

`object`	An object of class trial_sequence.
`p_control`	Probability of selecting a control, `NULL` for no sampling (default).
`period`	An integerish vector of non-zero length to select trial period(s) or `NULL` (default) to select all trial periods.
`subset_condition`	A string or `NULL` (default). `subset_condition` will be translated to a call (in case the expanded data is saved as a data.table or in the csv format) or to a SQL-query (in case the expanded data is saved as a duckdb file). The operators `"=="`, `"!="`, `">"`, `">="`, `"<"`, `"<="`, `%in%`, `"&"`, `"|"` are supported. Numeric vectors can be written as `c(1, 2, 3)` or `1:3`. Variables are not supported. *Note*: Make sure numeric vectors written as `1:3` are surrounded by spaces, e.g. `a %in% c( 1:4 , 6:9 )`, otherwise the code will fail.
`seed`	An integer seed or `NULL` (default). *Note*: The same seed will return a different result depending on the class of the te_datastore object contained in the trial_sequence object.

Details

This method is used on trial_sequence objects to read, subset and sample expanded data.

Value

An updated trial_sequence object, the data is stored in slot `@outcome_data` as a te_outcome_data object.

Examples

```
# create a trial_sequence-class object
trial_itt_dir <- file.path(tempdir(), "trial_itt")
dir.create(trial_itt_dir)
trial_itt <- trial_sequence(estimand = "ITT") |>
  set_data(data = data_censored) |>
  set_outcome_model(adjustment_terms = ~ x1 + x2)

trial_itt_csv <- set_expansion_options(
  trial_itt,
  output = save_to_csv(file.path(trial_itt_dir, "trial_csvs")),
  chunk_size = 500
) |>
  expand_trials()

# load_expanded_data default behaviour returns all trial_periods and doesn't sample
load_expanded_data(trial_itt_csv)

# load_expanded_data can subset the data before sampling
load_expanded_data(
  trial_itt_csv,
  p_control = 0.2,
  period = 1:20,
  subset_condition = "followup_time %in% 1:20 & x2 < 1",
)

# delete after use
unlink(trial_itt_dir, recursive = TRUE)
```

`outcome_data` *Outcome Data Accessor and Setter*

Description

[Experimental]

Usage

```
outcome_data(object)

outcome_data(object) <- value

## S4 method for signature 'trial_sequence'
outcome_data(object)

## S4 replacement method for signature 'trial_sequence'
outcome_data(object) <- value
```

Arguments

object	trial_sequence object
value	data.table to replace and update in @outcome_data

Details

Generic function to outcome data

Value

The object with updated outcome data

Examples

```
ts <- trial_sequence("ITT")
new_data <- data.table::data.table(vignette_switch_data[1:200, ])
new_data$weight <- 1
outcome_data(ts) <- new_data
```

---

parsnip_model *Fit outcome models using* parsnip *models*

Description

[Experimental]

Usage

```
parsnip_model(model_spec, save_path)
```

Arguments

model_spec	A parsnip model definition with mode = "classification".
save_path	Directory to save models. Set to NA if models should not be saved.

Details

Specify that the models should be fit using a classification model specified with the `parsnip` package.

Warning: This functionality is experimental and not recommended for use in analyses. $sqrt{n}$-consistency estimation and valid inference of the parameters in marginal structural models for emulated trials generally require that the weights for treatment switching and censoring be estimated at parametric rates, which is generally not possible when using data-adaptive estimation of high-dimensional regressions. Therefore, we only recommend using `stats_glm_logit()`.

Value

An object of class `te_parsnip_model` inheriting from te_model_fitter which is used for dispatching methods for the fitting models.

See Also

Other model_fitter: `stats_glm_logit()`, `te_model_fitter-class`

Examples

```
## Not run:
if (
  requireNamespace("parsnip", quietly = TRUE) &&
    requireNamespace("rpart", quietly = TRUE)
) {
  # Use a decision tree model fitted with the rpart package
  parsnip_model(
    model_spec = parsnip::decision_tree(tree_depth = 30) |>
      set_mode("classification") |>
      set_engine("rpart"),
    save_path = tempdir()
  )
}

## End(Not run)
```

---

predict_marginal *Predict marginal cumulative incidences with confidence intervals for a target trial population*

Description

[Stable] This function predicts the marginal cumulative incidences when a target trial population receives either the treatment or non-treatment at baseline (for an intention-to-treat analysis) or either sustained treatment or sustained non-treatment (for a per-protocol analysis). The difference between these cumulative incidences is the estimated causal effect of treatment. Currently, the `predict` function only provides marginal intention-to-treat and per-protocol effects, therefore it is only valid when `estimand_type = "ITT"` or `estimand_type = "PP"`.

Usage

```
predict(object, ...)

## S4 method for signature 'trial_sequence_ITT'
predict(
  object,
  newdata,
  predict_times,
  conf_int = TRUE,
  samples = 100,
  type = c("cum_inc", "survival")
)

## S4 method for signature 'trial_sequence_PP'
predict(
  object,
  newdata,
  predict_times,
  conf_int = TRUE,
  samples = 100,
  type = c("cum_inc", "survival")
)

## S3 method for class 'TE_msm'
predict(
  object,
  newdata,
  predict_times,
  conf_int = TRUE,
  samples = 100,
  type = c("cum_inc", "survival"),
  ...
)
```

Arguments

object	Object from trial_msm() or initiators() or trial_sequence.
...	Further arguments passed to or from other methods.
newdata	Baseline trial data that characterise the target trial population that marginal cumulative incidences or survival probabilities are predicted for. newdata must have the same columns and formats of variables as in the fitted marginal structural model specified in trial_msm() or initiators(). If newdata contains rows with followup_time > 0 these will be removed.
predict_times	Specify the follow-up visits/times where the marginal cumulative incidences or survival probabilities are predicted.
conf_int	Construct the point-wise 95-percent confidence intervals of cumulative incidences for the target trial population under treatment and non-treatment and their

differences by simulating the parameters in the marginal structural model from a multivariate normal distribution with the mean equal to the marginal structural model parameter estimates and the variance equal to the estimated robust covariance matrix.

`samples` Number of samples used to construct the simulation-based confidence intervals.

`type` Specify cumulative incidences or survival probabilities to be predicted. Either cumulative incidence (`"cum_inc"`) or survival probability (`"survival"`).

Value

A list of three data frames containing the cumulative incidences for each of the assigned treatment options (treatment and non-treatment) and the difference between them.

Examples

```
# Prediction for initiators() or trial_msm() objects -----

# If necessary set the number of `data.table` threads
data.table::setDTthreads(2)

data("te_model_ex")
predicted_ci <- predict(te_model_ex, predict_times = 0:30, samples = 10)

# Plot the cumulative incidence curves under treatment and non-treatment
plot(predicted_ci[[1]]$followup_time, predicted_ci[[1]]$cum_inc,
  type = "l",
  xlab = "Follow-up Time", ylab = "Cumulative Incidence",
  ylim = c(0, 0.7)
)
lines(predicted_ci[[1]]$followup_time, predicted_ci[[1]]$`2.5%`, lty = 2)
lines(predicted_ci[[1]]$followup_time, predicted_ci[[1]]$`97.5%`, lty = 2)

lines(predicted_ci[[2]]$followup_time, predicted_ci[[2]]$cum_inc, type = "l", col = 2)
lines(predicted_ci[[2]]$followup_time, predicted_ci[[2]]$`2.5%`, lty = 2, col = 2)
lines(predicted_ci[[2]]$followup_time, predicted_ci[[2]]$`97.5%`, lty = 2, col = 2)
legend("topleft", title = "Assigned Treatment", legend = c("0", "1"), col = 1:2, lty = 1)

# Plot the difference in cumulative incidence over follow up
plot(predicted_ci[[3]]$followup_time, predicted_ci[[3]]$cum_inc_diff,
  type = "l",
  xlab = "Follow-up Time", ylab = "Difference in Cumulative Incidence",
  ylim = c(0.0, 0.5)
)
lines(predicted_ci[[3]]$followup_time, predicted_ci[[3]]$`2.5%`, lty = 2)
lines(predicted_ci[[3]]$followup_time, predicted_ci[[3]]$`97.5%`, lty = 2)
```

print.TE_weight_summary *Print a weight summary object*

Description

[Stable]

Usage

```
## S3 method for class 'TE_weight_summary'
print(x, full = TRUE, ...)
```

Arguments

x	print TE_weight_summary object.
full	Print full or short summary.
...	Arguments passed to print.data.frame.

Value

No return value, only for printing.

read_expanded_data *Method to read expanded data*

Description

This method is used on te_datastore objects to read selected data and return one data.table.

Usage

```
read_expanded_data(object, period = NULL, subset_condition = NULL)

## S4 method for signature 'te_datastore_datatable'
read_expanded_data(object, period = NULL, subset_condition = NULL)
```

Arguments

object	An object of class te_datastore.
period	An integerish vector of non-zero length to select trial period(s) or NULL (default) to select all files.
subset_condition	A string of length 1 or NULL (default).

Value

A `data.frame` of class `data.table`.

Examples

```
# create a te_datastore_csv object and save some data
temp_dir <- tempfile("csv_dir_")
dir.create(temp_dir)
datastore <- save_to_csv(temp_dir)
data(vignette_switch_data)
expanded_csv_data <- save_expanded_data(datastore, vignette_switch_data[1:200, ])

# read expanded data
read_expanded_data(expanded_csv_data)

# delete after use
unlink(temp_dir, recursive = TRUE)
```

`sample_expanded_data` *Internal method to sample expanded data*

Description

Internal method to sample expanded data

Usage

```
sample_expanded_data(
  object,
  p_control,
  period = NULL,
  subset_condition = NULL,
  seed
)

## S4 method for signature 'te_datastore'
sample_expanded_data(
  object,
  p_control,
  period = NULL,
  subset_condition = NULL,
  seed
)
```

Arguments

object	An object of class te_datastore.
p_control	Probability of selecting a control.
period	An integerish vector of non-zero length to select trial period(s) or NULL (default) to select all trial periods.
subset_condition	A string or NULL.
seed	An integer seed or NULL (default).

Value

A data.frame of class data.table.

Examples

```
# Data object normally created by [expand_trials]
datastore <- new("te_datastore_datatable", data = te_data_ex$data, N = 50139L)

sample_expanded_data(datastore, period = 260:275, p_control = 0.2, seed = 123)
```

---

save_expanded_data *Method to save expanded data*

Description

This method is used internally by expand_trials to save the data to the "datastore" defined in set_expansion_options.

Usage

```
save_expanded_data(object, data)

## S4 method for signature 'te_datastore_datatable'
save_expanded_data(object, data)
```

Arguments

object	An object of class te_datastore or a child class.
data	A data frame containing the expanded trial data. The columns trial_period and id are present, which may be used in methods to save the data in an optimal way, such as with indexes, keys or separate files.

Value

An updated object with the data stored. Notably object@N should be increased

Examples

```
temp_dir <- tempfile("csv_dir_")
dir.create(temp_dir)
datastore <- save_to_csv(temp_dir)
data(vignette_switch_data)
save_expanded_data(datastore, vignette_switch_data[1:200, ])

# delete after use
unlink(temp_dir, recursive = TRUE)
```

`save_to_csv` *Save expanded data as CSV*

Description

[Experimental]

Usage

```
save_to_csv(path)
```

Arguments

`path`	Directory to save CSV files in. Must be empty.

Value

A te_datastore_csv object.

See Also

Other save_to: `save_to_datatable()`, `save_to_duckdb()`, `set_expansion_options()`

Examples

```
csv_dir <- file.path(tempdir(), "expanded_trials_csv")
dir.create(csv_dir)
csv_datastore <- save_to_csv(path = csv_dir)

trial_to_expand <- trial_sequence("ITT") |>
  set_data(data = data_censored) |>
  set_expansion_options(output = csv_datastore, chunk_size = 500)

# Delete directory after use
unlink(csv_dir)
```

save_to_datatable *Save expanded data as a* data.table

Description

[Experimental]

Usage

```
save_to_datatable()
```

See Also

Other save_to: save_to_csv(), save_to_duckdb(), set_expansion_options()

Examples

```
trial_to_expand <- trial_sequence("ITT") |>
  set_data(data = data_censored) |>
  set_expansion_options(output = save_to_datatable(), chunk_size = 500)
```

save_to_duckdb *Save expanded data to* DuckDB

Description

[Experimental]

Usage

```
save_to_duckdb(path)
```

Arguments

path	Directory to save DuckDB database file in.

Value

A te_datastore_duckdb object.

See Also

Other save_to: save_to_csv(), save_to_datatable(), set_expansion_options()

Examples

```
if (require(duckdb)) {
  duckdb_dir <- file.path(tempdir(), "expanded_trials_duckdb")

  trial_to_expand <- trial_sequence("ITT") |>
    set_data(data = data_censored) |>
    set_expansion_options(output = save_to_duckdb(path = duckdb_dir), chunk_size = 500)

  # Delete directory after use
  unlink(duckdb_dir)
}
```

---

set_censor_weight_model

Set censoring weight model

Description

[Experimental]

Usage

```
set_censor_weight_model(
  object,
  censor_event,
  numerator,
  denominator,
  pool_models = NULL,
  model_fitter
)

## S4 method for signature 'trial_sequence'
set_censor_weight_model(
  object,
  censor_event,
  numerator,
  denominator,
  pool_models = c("none", "both", "numerator"),
  model_fitter = stats_glm_logit()
)

## S4 method for signature 'trial_sequence_PP'
set_censor_weight_model(
  object,
  censor_event,
  numerator,
```

```
  denominator,
  pool_models = "none",
  model_fitter = stats_glm_logit()
)

## S4 method for signature 'trial_sequence_ITT'
set_censor_weight_model(
  object,
  censor_event,
  numerator,
  denominator,
  pool_models = "numerator",
  model_fitter = stats_glm_logit()
)

## S4 method for signature 'trial_sequence_AT'
set_censor_weight_model(
  object,
  censor_event,
  numerator,
  denominator,
  pool_models = "none",
  model_fitter = stats_glm_logit()
)
```

Arguments

`object`	trial_sequence.
`censor_event`	string. Name of column containing censoring indicator.
`numerator`	A RHS formula to specify the logistic models for estimating the numerator terms of the inverse probability of censoring weights.
`denominator`	A RHS formula to specify the logistic models for estimating the denominator terms of the inverse probability of censoring weights.
`pool_models`	Fit pooled or separate censoring models for those treated and those untreated at the immediately previous visit. Pooling can be specified for the models for the numerator and denominator terms of the inverse probability of censoring weights. One of "none", "numerator", or "both" (default is "none" except when estimand = "ITT" then default is "numerator").
`model_fitter`	An object of class `te_model_fitter` which determines the method used for fitting the weight models. For logistic regression use `stats_glm_logit()`.

Value

`object` is returned with `@censor_weights` set

Examples

```
trial_sequence("ITT") |>
```

```
  set_data(data = data_censored) |>
  set_censor_weight_model(
    censor_event = "censored",
    numerator = ~ age_s + x1 + x3,
    denominator = ~ x3 + x4,
    pool_models = "both",
    model_fitter = stats_glm_logit(save_path = tempdir())
  )
```

---

set_data *Set the trial data*

Description

[Experimental]

Usage

```
set_data(object, data, ...)

## S4 method for signature 'trial_sequence_ITT,data.frame'
set_data(
  object,
  data,
  id = "id",
  period = "period",
  treatment = "treatment",
  outcome = "outcome",
  eligible = "eligible"
)

## S4 method for signature 'trial_sequence_AT,data.frame'
set_data(
  object,
  data,
  id = "id",
  period = "period",
  treatment = "treatment",
  outcome = "outcome",
  eligible = "eligible"
)

## S4 method for signature 'trial_sequence_PP,data.frame'
set_data(
  object,
  data,
  id = "id",
  period = "period",
```

```
  treatment = "treatment",
  outcome = "outcome",
  eligible = "eligible"
)
```

Arguments

object	A trial_sequence object
data	A data.frame containing all the required variables in the person-time format, i.e., the <U+2018>long<U+2019> format.
...	Other arguments used by methods internally.
id	Name of the variable for identifiers of the individuals. Default is <U+2018>id<U+2019>.
period	Name of the variable for the visit/period. Default is <U+2018>period<U+2019>.
treatment	Name of the variable for the treatment indicator at that visit/period. Default is <U+2018>treatment<U+2019>.
outcome	Name of the variable for the indicator of the outcome event at that visit/period. Default is <U+2018>outcome<U+2019>.
eligible	Name of the variable for the indicator of eligibility for the target trial at that visit/period. Default is <U+2018>eligible<U+2019>.

Value

An updated trial_sequence object with data

Examples

```
data(trial_example)
trial_sequence("ITT") |>
  set_data(
    data = trial_example,
    id = "id",
    period = "period",
    eligible = "eligible",
    treatment = "treatment"
  )
```

---

set_expansion_options *Set expansion options*

Description

[Experimental]

Usage

```
set_expansion_options(object, ...)

## S4 method for signature 'trial_sequence_ITT'
set_expansion_options(
  object,
  output,
  chunk_size,
  first_period = 0,
  last_period = Inf
)

## S4 method for signature 'trial_sequence_PP'
set_expansion_options(
  object,
  output,
  chunk_size,
  first_period = 0,
  last_period = Inf
)

## S4 method for signature 'trial_sequence_ITT'
set_expansion_options(
  object,
  output,
  chunk_size,
  first_period = 0,
  last_period = Inf
)
```

Arguments

object	A trial_sequence object
...	Arguments used in methods
output	A te_datastore object as created by a save_to_* function.
chunk_size	An integer specifying the number of patients to include in each expansion iteration
first_period	An integer specifying the first period to include in the expansion
last_period	An integer specifying the last period to include in the expansion

Value

object is returned with @expansion set

See Also

Other save_to: save_to_csv(), save_to_datatable(), save_to_duckdb()

Examples

```
output_dir <- file.path(tempdir(check = TRUE), "expanded_data")
ITT_trial <- trial_sequence("ITT") |>
  set_data(data = data_censored) |>
  set_expansion_options(output = save_to_csv(output_dir), chunk_size = 500)

# Delete directory
unlink(output_dir, recursive = TRUE)
```

---

set_outcome_model *Specify the outcome model*

Description

[Experimental]

The time-to-event model for `outcome` is specified with this method. Any adjustment terms can be specified. For ITT and PP estimands the `treatment_var` is not specified as it is automatically defined as `assigned_treatment`. Importantly, the modelling of "time" is specified in this model with arguments for trial start time and follow up time within the trial.

Usage

```
set_outcome_model(object, ...)

## S4 method for signature 'trial_sequence'
set_outcome_model(
  object,
  treatment_var = ~0,
  adjustment_terms = ~1,
  followup_time_terms = ~followup_time + I(followup_time^2),
  trial_period_terms = ~trial_period + I(trial_period^2),
  model_fitter = stats_glm_logit(save_path = NA)
)

## S4 method for signature 'trial_sequence_ITT'
set_outcome_model(
  object,
  adjustment_terms = ~1,
  followup_time_terms = ~followup_time + I(followup_time^2),
  trial_period_terms = ~trial_period + I(trial_period^2),
  model_fitter = stats_glm_logit(save_path = NA)
)

## S4 method for signature 'trial_sequence_PP'
set_outcome_model(
  object,
  adjustment_terms = ~1,
```

```
  followup_time_terms = ~followup_time + I(followup_time^2),
  trial_period_terms = ~trial_period + I(trial_period^2),
  model_fitter = stats_glm_logit(save_path = NA)
)

## S4 method for signature 'trial_sequence_AT'
set_outcome_model(
  object,
  treatment_var = "dose",
  adjustment_terms = ~1,
  followup_time_terms = ~followup_time + I(followup_time^2),
  trial_period_terms = ~trial_period + I(trial_period^2),
  model_fitter = stats_glm_logit(save_path = NA)
)
```

Arguments

`object`	A trial_sequence object
`...`	Parameters used by methods
`treatment_var`	The treatment term, only used for "as treated" estimands. PP and ITT are fixed to use "assigned_treatment".
`adjustment_terms`	Formula terms for any covariates to adjust the outcome model.
`followup_time_terms`	Formula terms for `followup_time`, the time period relative to the start of the trial.
`trial_period_terms`	Formula terms for `trial_period`, the time period of the start of the trial.
`model_fitter`	A `te_model_fitter` object, e.g. from `stats_glm_logit()`.

Value

A modified `object` with the `outcome_model` slot set

Examples

```
trial_sequence("ITT") |>
  set_data(data_censored) |>
  set_outcome_model(
    adjustment_terms = ~age_s,
    followup_time_terms = ~ stats::poly(followup_time, degree = 2)
  )
```

set_switch_weight_model *Set switching weight model*

Description

[Experimental]

Usage

```
set_switch_weight_model(object, numerator, denominator, model_fitter, ...)

## S4 method for signature 'trial_sequence'
set_switch_weight_model(
  object,
  numerator,
  denominator,
  model_fitter,
  eligible_wts_0 = NULL,
  eligible_wts_1 = NULL
)

## S4 method for signature 'trial_sequence_ITT'
set_switch_weight_model(object, numerator, denominator, model_fitter)
```

Arguments

object	A trial_sequence object.
numerator	Right hand side formula for the numerator model
denominator	Right hand side formula for the denominator model
model_fitter	A te_model_fitter object, such as stats_glm_logit
...	Other arguments used by methods.
eligible_wts_0	Name of column containing indicator (0/1) for observation to be excluded/included in weight model.
eligible_wts_1	Exclude some observations when fitting the models for the inverse probability of treatment weights. For example, if it is assumed that an individual will stay on treatment for at least 2 visits, the first 2 visits after treatment initiation by definition have a probability of staying on the treatment of 1.0 and should thus be excluded from the weight models for those who are on treatment at the immediately previous visit. Users can define a variable that indicates that these 2 observations are ineligible for the weight model for those who are on treatment at the immediately previous visit and add the variable name in the argument eligible_wts_1. Similar definitions are applied to eligible_wts_0 for excluding observations when fitting the models for the inverse probability of treatment weights for those who are not on treatment at the immediately previous visit.

Value

`object` is returned with `@switch_weights` set

Examples

```
trial_sequence("PP") |>
  set_data(data = data_censored) |>
  set_switch_weight_model(
    numerator = ~ age_s + x1 + x3,
    denominator = ~ x3 + x4,
    model_fitter = stats_glm_logit(tempdir())
  )
```

---

show_weight_models *Show Weight Model Summaries*

Description

[Experimental]

Usage

```
show_weight_models(object)
```

Arguments

`object`	A trial_sequence object after fitting weight models with `calculate_weights()`

Value

Prints summaries of the censoring models

---

stats_glm_logit *Fit outcome models using* `stats::glm`

Description

[Experimental]

Usage

```
stats_glm_logit(save_path)
```

Arguments

`save_path`	Directory to save models. Set to `NA` if models should not be saved.

Details

Specify that the pooled logistic regression outcome models should be fit using stats::glm with `family = binomial(link = "logit")`.

Outcome models additional calculate robust variance estimates using `sandwich::vcovCL`.

Value

An object of class `te_stats_glm_logit` inheriting from te_model_fitter which is used for dispatching methods for the fitting models.

See Also

Other model_fitter: `parsnip_model()`, `te_model_fitter-class`

Examples

```
stats_glm_logit(save_path = tempdir())
```

---

summary.TE_data_prep *Summary methods*

Description

[Stable] Print summaries of data and model objects produced by `TrialEmulation`.

Usage

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

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

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

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

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

Arguments

`object`	Object to print summary
`...`	Additional arguments passed to print methods.

Value

No value, displays summaries of object.

`te_data-class` *TrialEmulation Data Class*

Description

TrialEmulation Data Class

Slots

`data` A `data.table` object with columns "id", "period", "treatment", "outcome", "eligible"

`te_datastore-class` *te_datastore*

Description

This is the parent class for classes which define how the expanded trial data should be stored. To define a new storage type, a new class should be defined which inherits from `te_datastore`. In addition, methods save_expanded_data and `read_expanded_data` need to be defined for the new class.

Value

A 'te_datastore' object

Slots

`N` The number of observations in this data. Initially 0.

te_data_ex *Example of a prepared data object*

Description

A small example object from data_preparation used in examples. It is created with the following code:

Usage

```
te_data_ex
```

Format

An object of class `TE_data_prep_dt` (inherits from `TE_data_prep`) of length 6.

Details

```
dat <- trial_example[trial_example$id < 200, ]

te_data_ex <- data_preparation(
data = dat,
 outcome_cov = c("nvarA", "catvarA"),
 first_period = 260,
 last_period = 280
)
```

See Also

te_model_ex

te_model_ex *Example of a fitted marginal structural model object*

Description

A small example object from trial_msm used in examples. It is created with the following code:

Usage

```
te_model_ex
```

Format

An object of class `TE_msm` of length 3.

Details

```
te_model_ex <- trial_msm(
 data = data_subset,
 outcome_cov = c("catvarA", "nvarA"),
 last_followup = 40,
 model_var = "assigned_treatment",
 include_followup_time = ~followup_time,
 include_trial_period = ~trial_period,
 use_sample_weights = FALSE,
 quiet = TRUE,
 glm_function = "glm"
 )
```

See Also

te_data_ex

---

te_model_fitter-class *Outcome Model Fitter Class*

Description

This is a virtual class which other outcome model fitter classes should inherit from. Objects of these class exist to define how the outcome models are fit. They are used for the dispatch of the internal methods fit_outcome_model, fit_weights_model and predict.

See Also

Other model_fitter: `parsnip_model()`, `stats_glm_logit()`

---

te_outcome_data-class *TrialEmulation Outcome Data Class*

Description

TrialEmulation Outcome Data Class

Slots

`data` A `data.table` object with columns "id", "period",

`n_rows` Number of rows

`n_ids` Number of IDs

`periods` Vector of periods "treatment", "outcome", "eligible"

te_outcome_fitted-class

Fitted Outcome Model Object

Description

Fitted Outcome Model Object

Slots

`model` list containing fitted model objects.

`summary` list of data.frames. Tidy model summaries a la `broom()` and `glance()`

te_outcome_model-class

Fitted Outcome Model Object

Description

Fitted Outcome Model Object

Slots

`formula` `formula` object for the model fitting

`adjustment_vars` character. Adjustment variables

`treatment_var` Variable used for treatment

`stabilised_weights_terms` formula. Adjustment terms from numerator models of stabilised weights. These must be included in the outcome model.

`adjustment_terms` formula. User specified terms to include in the outcome model

`treatment_terms` formula. Estimand defined treatment term

`followup_time_terms` formula. Terms to model follow up time within an emulated trial

`trial_period_terms` formula. Terms to model start time ("trial_period") of an emulated trial

`model_fitter` Model fitter object

`fitted` list. Saves the model objects

trial_example *Example of longitudinal data for sequential trial emulation*

Description

A dataset containing the treatment, outcomes and other attributes of 503 patients for sequential trial emulation. See `vignette("Getting-Started")`.

Usage

```
trial_example
```

Format

A data frame with 48400 rows and 11 variables:

id patient identifier

eligible eligible for trial start in this period, 1=yes, 0=no

period time period

outcome indicator for outcome in this period, 1=event occurred, 0=no event

treatment indicator for receiving treatment in this period, 1=treatment, 0=no treatment

catvarA A categorical variable relating to treatment and the outcome

catvarB A categorical variable relating to treatment and the outcome

catvarC A categorical variable relating to treatment and the outcome

nvarA A numerical variable relating to treatment and the outcome

nvarB A numerical variable relating to treatment and the outcome

nvarC A numerical variable relating to treatment and the outcome

trial_msm *Fit the marginal structural model for the sequence of emulated trials*

Description

[Stable]

Usage

```
trial_msm(
  data,
  outcome_cov = ~1,
  estimand_type = c("ITT", "PP", "As-Treated"),
  model_var = NULL,
  first_followup = NA,
  last_followup = NA,
  analysis_weights = c("asis", "unweighted", "p99", "weight_limits"),
  weight_limits = c(0, Inf),
  include_followup_time = ~followup_time + I(followup_time^2),
  include_trial_period = ~trial_period + I(trial_period^2),
  where_case = NA,
  glm_function = c("glm", "parglm"),
  use_sample_weights = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

data	A data.frame containing all the required variables in the person-time format, i.e., the 'long' format.
outcome_cov	A RHS formula with baseline covariates to be adjusted for in the marginal structural model for the emulated trials. Note that if a time-varying covariate is specified in outcome_cov, only its value at each of the trial baselines will be included in the expanded data.
estimand_type	Specify the estimand for the causal analyses in the sequence of emulated trials. estimand_type = "ITT" will perform intention-to-treat analyses, where treatment switching after trial baselines are ignored. estimand_type = "PP" will perform per-protocol analyses, where individuals' follow-ups are artificially censored and inverse probability of treatment weighting is applied. estimand_type = "As-Treated" will fit a standard marginal structural model for all possible treatment sequences, where individuals' follow-ups are not artificially censored but treatment switching after trial baselines are accounted for by applying inverse probability of treatment weighting.
model_var	Treatment variables to be included in the marginal structural model for the emulated trials. model_var = "assigned_treatment" will create a variable assigned_treatment that is the assigned treatment at the trial baseline, typically used for ITT and per-protocol analyses. model_var = "dose" will create a variable dose that is the cumulative number of treatments received since the trial baseline, typically used in as-treated analyses.
first_followup	First follow-up time/visit in the trials to be included in the marginal structural model for the outcome event.
last_followup	Last follow-up time/visit in the trials to be included in the marginal structural model for the outcome event.

analysis_weights
: Choose which type of weights to be used for fitting the marginal structural model for the outcome event.
  - "asis": use the weights as calculated.
  - "p99": use weights truncated at the 1st and 99th percentiles (based on the distribution of weights in the entire sample).
  - "weight_limits": use weights truncated at the values specified in weight_limits.
  - "unweighted": set all analysis weights to 1, even if treatment weights or censoring weights were calculated.

weight_limits
: Lower and upper limits to truncate weights, given as c(lower, upper)

include_followup_time
: The model to include the follow up time/visit of the trial (followup_time) in the marginal structural model, specified as a RHS formula.

include_trial_period
: The model to include the trial period (trial_period) in the marginal structural model, specified as a RHS formula.

where_case
: Define conditions using variables specified in where_var when fitting a marginal structural model for a subgroup of the individuals. For example, if where_var= "age", where_case = "age >= 30" will only fit the marginal structural model to the subgroup of individuals. who are 30 years old or above.

glm_function
: Specify which glm function to use for the marginal structural model from the stats or parglm packages. The default function is the glm function in the stats package. Users can also specify glm_function = "parglm" such that the parglm function in the parglm package can be used for fitting generalized linear models in parallel. The default control setting for parglm is nthreads = 4 and method = "FAST", where four cores and Fisher information are used for faster computation. Users can change the default control setting by passing the arguments nthreads and method in the parglm.control function of the parglm package, or alternatively, by passing a control argument with a list produced by parglm.control(nthreads = , method =).

use_sample_weights
: Use case-control sampling weights in addition to inverse probability weights for treatment and censoring. data must contain a column sample_weight. The final weights used in the pooled logistic regression are calculated as weight = weight * sample_weight.

quiet
: Suppress the printing of progress messages and summaries of the fitted models.

...
: Additional arguments passed to glm_function. This may be used to specify initial values of parameters or arguments to control. See stats::glm, parglm::parglm and parglm::parglm.control() for more information.

Details

Apply a weighted pooled logistic regression to fit the marginal structural model for the sequence of emulated trials and calculates the robust covariance matrix of parameter using the sandwich estimator.

The model formula is constructed by combining the arguments outcome_cov, model_var, include_followup_time, and include_trial_period.

Value

Object of class `TE_msm` containing

model a `glm` object

robust a list containing a summary table of estimated regression coefficients and the robust covariance matrix

args a list contain the parameters used to prepare and fit the model

---

`trial_sequence` *Create a sequence of emulated target trials object*

Description

[Experimental]

Usage

```
trial_sequence(estimand, ...)
```

Arguments

`estimand`	The name of the estimand for this analysis, either one of `"ITT"`, `"PP"`, `"AT"` for intention-to-treat, per-protocol, as-treated estimands respectively, or the name of a class extending trial_sequence
`...`	Other parameters used when creating object

Value

An estimand specific trial sequence object

Examples

```
trial_sequence("ITT")
```

trial_sequence-class *Trial Sequence class*

Description

Trial Sequence class

Slots

`data` te_data.

`estimand` character. Descriptive name of estimand.

`expansion` te_expansion

`outcome_model` te_outcome_model.

`outcome_data` te_outcome_data.

`censor_weight` te_weight. Object to define weighting to account for informative censoring

`censor_weight` te_weight. Object to define weighting to account for informative censoring due to treatment switching

vignette_switch_data *Example of expanded longitudinal data for sequential trial emulation*

Description

This is the expanded dataset created in the `vignette("Getting-Started")` known as `switch_data`.

Usage

```
vignette_switch_data
```

Format

A data frame with 1939053 rows and 7 variables:

id patient identifier

trial_period trial start time period

followup_time follow up time within trial

outcome indicator for outcome in this period, 1=event occurred, 0=no event

treatment indicator for receiving treatment in this period, 1=treatment, 0=non-treatment

assigned_treatment indicator for assigned treatment at baseline of the trial, 1=treatment, 0=non-treatment

weight weights for use with model fitting

catvarA A categorical variable relating to treatment and the outcome

catvarB A categorical variable relating to treatment and the outcome

catvarC A categorical variable relating to treatment and the outcome

nvarA A numerical variable relating to treatment and the outcome

nvarB A numerical variable relating to treatment and the outcome

nvarC A numerical variable relating to treatment and the outcome

---

`weight_model_data_indices` *Data used in weight model fitting*

Description

[Experimental]

Usage

```
weight_model_data_indices(
  object,
  type = c("switch", "censor"),
  model,
  set_col = NULL
)
```

Arguments

`object`	A trial_sequence object
`type`	Select a censoring or switching model
`model`	The model name
`set_col`	A character string to specifying a new column to contain indicators for observations used in fitting this model.

Value

If `set_col` is not specified a logical `data.table` column is returned. Otherwise

Examples

```
trial_pp <- trial_sequence("PP") |>
  set_data(data_censored) |>
  set_switch_weight_model(
    numerator = ~age,
    denominator = ~ age + x1 + x3,
    model_fitter = stats_glm_logit(tempdir())
  ) |>
  calculate_weights()
ipw_data(trial_pp)
```

```
show_weight_models(trial_pp)

# get logical column for own processing
i <- weight_model_data_indices(trial_pp, "switch", "d0")

# set column in data
weight_model_data_indices(trial_pp, "switch", "d0", set_col = "sw_d0")
weight_model_data_indices(trial_pp, "switch", "d1", set_col = "sw_d1")
ipw_data(trial_pp)
```

Index