Package 'MRTAnalysis'

September 9, 2025

Type Package

Title Assessing Proximal, Distal, and Mediated Causal Excursion Effects for Micro-Randomized Trials

Version 0.3.0

Description Provides methods to analyze micro-randomized trials (MRTs) with binary treatment options. Supports three types of analyses: (1) proximal causal excursion effects, including weighted and centered least squares (WCLS) for continuous proximal outcomes by Boruvka et al. (2018) <doi:10.1080/01621459.2017.1305274> and the estimator for marginal excursion effect (EMEE) for binary proximal outcomes by Qian et al. (2021) <doi:10.1093/biomet/asaa070>; (2) distal causal excursion effects (DCEE) for continuous distal outcomes using a two-stage estimator by Qian (2025) <doi:10.48550/arXiv.2502.13500>; and (3) mediated causal excursion effects (MCEE) for continuous distal outcomes, estimating natural direct and indirect excursion effects in the presence of time-varying mediators by Qian (2025) <doi:10.48550/arXiv.2506.20027>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports rootSolve, stats, geepack, sandwich, mgcv, randomForest, ranger

Depends R (>= 4.2)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), SuperLearner, earth, dplyr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Tianchen Qian [aut, cre] (ORCID: <https://orcid.org/0000-0003-4282-7826>), Shaolin Xiang [aut], Zhaoxi Cheng [aut], Audrey Boruvka [ctb]

Maintainer Tianchen Qian <t.qian@uci.edu>

Repository CRAN

Date/Publication 2025-09-09 07:20:31 UTC

Contents

`.mcee_assert_df` *Assert that input is a data frame*

Description

Assert that input is a data frame

Usage

```
.mcee_assert_df(data)
```

Arguments

`data`	Object to check

Value

Invisibly TRUE if valid, otherwise stops with error

`.mcee_build_f_matrix` *Build basis matrix f(t) from time-varying effect formula*

Description

Build basis matrix f(t) from time-varying effect formula

Usage

```
.mcee_build_f_matrix(time_varying_effect_form, data)
```

Arguments

time_varying_effect_form	RHS-only formula
data	Data frame to evaluate formula on

Value

Model matrix with basis functions evaluated at each row

.mcee_build_weights *Build per-row weights omega(i,t) for MCEE estimation*

Description

Build per-row weights omega(i,t) for MCEE estimation

Usage

```
.mcee_build_weights(
  data,
  id,
  dp,
  weight_per_row = NULL,
  specific_dp_only = NULL,
  verbose = TRUE
)
```

Arguments

data	Data frame
id	Subject ID column name
dp	Decision point column name
weight_per_row	Optional user-specified per-row weights
specific_dp_only	Optional vector of decision points to target (others get weight 0)
verbose	Logical; whether to print informative messages

Value

Numeric vector of per-row weights

.mcee_check_binary_col

Validate binary column coding

Description

Validate binary column coding

Usage

```
.mcee_check_binary_col(data, col, allow_all1 = TRUE, label = NULL)
```

Arguments

data	Data frame containing the column
col	Column name (can be NULL, in which case validation is skipped)
allow_all1	Logical; if FALSE, column cannot be all 1s
label	Optional descriptive label for error messages

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_check_control_formula

Validate control formula excludes treatment and outcome

Description

Validate control formula excludes treatment and outcome

Usage

```
.mcee_check_control_formula(control_formula, treatment, outcome, dp, label)
```

Arguments

control_formula	RHS-only formula for control variables
treatment	Treatment column name
outcome	Outcome column name
dp	Decision point column name (used in error messages)
label	Descriptive label for error messages

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_check_dp_strictly_increasing *Check that decision points are strictly increasing within each subject*

Description

Check that decision points are strictly increasing within each subject

Usage

```
.mcee_check_dp_strictly_increasing(data, id, dp)
```

Arguments

data	Data frame
id	Column name for subject ID
dp	Column name for decision point

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_check_formula_mediator *Check config formula for inclusion/exclusion of mediator*

Description

Check config formula for inclusion/exclusion of mediator

Usage

```
.mcee_check_formula_mediator(config, target, mediator)
```

Arguments

config	A nuisance config list (may or may not contain ‘formula‘).
target	Character scalar, one of "p", "q", "eta", "mu", "nu".
mediator	Character scalar: mediator variable name.

Value

Invisibly TRUE. Warnings are produced if formulas look suspicious.

.mcee_check_id_rows_grouped

Check that rows for each subject appear in contiguous blocks

Description

Check that rows for each subject appear in contiguous blocks

Usage

```
.mcee_check_id_rows_grouped(data, id, max_show = 5)
```

Arguments

data	Data frame
id	Column name for subject ID
max_show	Maximum number of offending IDs to show in error message

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_check_no_missing_vars

Check data frame columns for missing/infinite values

Description

Check data frame columns for missing/infinite values

Usage

```
.mcee_check_no_missing_vars(data, vars, where = NULL, max_show = 5)
```

Arguments

data	Data frame to check
vars	Character vector of column names to check
where	Optional context description for error messages
max_show	Maximum number of row indices to show per variable

Value

Invisibly TRUE if no missing data found, otherwise stops with error

---

`.mcee_check_no_missing_vec` *Check numeric vector for missing/infinite values*

Description

Check numeric vector for missing/infinite values

Usage

```
.mcee_check_no_missing_vec(vec, name, max_show = 5)
```

Arguments

`vec`	Numeric vector to check
`name`	Variable name for error messages
`max_show`	Maximum number of row indices to show

Value

Invisibly TRUE if no missing data found, otherwise stops with error

---

`.mcee_check_outcome_constant_within_id` *Check that outcome is constant within each subject (required for distal outcomes)*

Description

Check that outcome is constant within each subject (required for distal outcomes)

Usage

```
.mcee_check_outcome_constant_within_id(data, id, outcome)
```

Arguments

`data`	Data frame
`id`	Column name for subject ID
`outcome`	Column name for outcome

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_check_time_varying_effect_form

Validate time-varying effect formula structure

Description

Validate time-varying effect formula structure

Usage

```
.mcee_check_time_varying_effect_form(time_varying_effect_form, dp)
```

Arguments

time_varying_effect_form	RHS-only formula for basis functions
dp	Decision point column name

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_compact_model_info

Generate compact one-line description of nuisance model object

Description

Generate compact one-line description of nuisance model object

Usage

```
.mcee_compact_model_info(obj)
```

Arguments

obj	Fitted model object or known value descriptor

Value

Character string describing the object

`.mcee_core_rows` *Numerical core implementing MCEE estimation mathematics*

Description

Implements the core MCEE estimating equations and sandwich variance estimation. This function contains the mathematical heart of the MCEE method, solving the weighted estimating equations for α (NDEE) and β (NIEE).

Usage

```
.mcee_core_rows(
  n,
  f_nrows,
  omega_nrows,
  i_index,
  phi11_vec,
  phi10_vec,
  phi00_vec
)
```

Arguments

`n`	Integer. Number of unique subjects.
`f_nrows`	Matrix $nrows \times p$. Row `r` contains $f(t_r)^T$, the basis functions evaluated at the decision point for row `r`.
`omega_nrows`	Numeric vector of length `nrows`. Per-row weights $\omega(i,t)$.
`i_index`	Integer vector of length `nrows`. Subject index (1 to `n`) for each row, indicating which subject row `r` belongs to.
`phi11_vec`, `phi10_vec`, `phi00_vec`	Numeric vectors of length `nrows`. Influence function values for each row, computed from nuisance predictions.

Details

MCEE Estimating Equations:

- **NDEE**: $\alpha = S^{-1} \times (1/n) \sum_{i,t} \omega(i,t)\{\phi_t^{10} - \phi_t^{00}\} f(t)$
- **NIEE**: $\beta = S^{-1} \times (1/n) \sum_{i,t} \omega(i,t)\{\phi_t^{11} - \phi_t^{10}\} f(t)$

where $S = (1/n) \sum_{i,t} \omega(i,t) f(t) f(t)^T$.

Sandwich Variance Formula: $\text{Var}((\alpha, \beta)) = \text{Bread}^{-1} \times \text{Meat} \times \text{Bread}^{-1,T}/n$, where:

- **Bread** = blockdiag(S, S) ($2p \times 2p$ matrix)
- **Meat** = $(1/n) \sum_i U_i U_i^T$, with subject-level score vectors: $U_i = \sum_t \omega(i,t) \times [\{\phi_t^{10} - \phi_t^{00} - f^T\alpha\} f; \{\phi_t^{11} - \phi_t^{10} - f^T\beta\} f]$

Mathematical Details: The implementation follows the theoretical framework detailed in the MCEE vignette appendix. The estimating equations are based on efficient influence functions for the causal parameters of interest in the mediation analysis setting.

Value

List containing:

`alpha_hat` Vector of length p: NDEE parameter estimates

`alpha_se` Vector of length p: NDEE standard errors

`beta_hat` Vector of length p: NIEE parameter estimates

`beta_se` Vector of length p: NIEE standard errors

`varcov` Matrix $2p \times 2p$: Joint variance-covariance for (α, β)

`alpha_varcov` Matrix $p \times p$: Variance-covariance for α only

`beta_varcov` Matrix $p \times p$: Variance-covariance for β only

`.mcee_default_family` *Select default GLM family based on nuisance parameter type*

Description

Select default GLM family based on nuisance parameter type

Usage

```
.mcee_default_family(target, method)
```

Arguments

`target`	Nuisance parameter name ("p", "q", "eta", "mu", "nu")
`method`	Learning method name

Value

GLM family object or NULL for non-GLM methods

.mcee_drop_var_from_rhs *Remove a variable from RHS-only formula*

Description

Remove a variable from RHS-only formula

Usage

```
.mcee_drop_var_from_rhs(rhs_only_formula, var)
```

Arguments

rhs_only_formula	RHS-only formula
var	Variable name to remove

Value

Modified formula with variable removed

.mcee_fit_nuisance *Fit a single nuisance component with flexible learner support*

Description

Internal workhorse function that fits individual nuisance parameters using various machine learning methods or known constants. Handles the complexity of different learner APIs and provides consistent predictions.

Usage

```
.mcee_fit_nuisance(
  config,
  data_for_fitting,
  data_for_predicting,
  lhs_var,
  param_name,
  data_for_fitting_name
)
```

Arguments

`config` Configuration list describing how to fit the nuisance parameter. Created by `mcee_config_maker` or helper functions. Contains:

- **Known values**: `known`, `known_a1`, `known_a0` (bypasses fitting)
- **Model-based**: `method`, `formula`, `family`, optional `clipping`
- **Method-specific**: `SL.library` for SuperLearner, learner-specific args

`data_for_fitting` Data frame subset used to train the model (e.g., available rows only).

`data_for_predicting` Data frame on which to generate predictions (usually full data).

`lhs_var` Character. Column name of the response/outcome variable to model.

`param_name` Character. Descriptive name for error messages (e.g., "p_t(1|H_t)").

`data_for_fitting_name` Character. Description of fitting data for model call display.

Details

Supported Methods:

- `"glm"`: Uses `stats::glm()` with automatic family detection
- `"lm"`: Uses `stats::lm()` (continuous outcomes only)
- `"gam"`: Uses `mgcv::gam()` supporting smooth terms
- `"rf"`: Uses `randomForest::randomForest()`
- `"ranger"`: Uses `ranger::ranger()` (faster random forest)
- `"sl"`: Uses `SuperLearner::SuperLearner()`

Automatic Family Detection: When `family=NULL` in GLM/GAM configs: - Binary outcomes (0/1 only): `binomial()` - Continuous outcomes: `gaussian()`

Known Values: If any of `known`, `known_a1`, `known_a0` is provided, no model is fitted. Returns constant predictions and a descriptor object.

Value

List with components:

`pred` Numeric vector of length `nrow(data_for_predicting)` containing predictions/fitted values.

`model` Fitted model object (e.g., `glm`, `gam`, `randomForest`) or a list descriptor for known values.

.mcee_message_if_no_availability_provided

Print informative message if no availability column provided

Description

Print informative message if no availability column provided

Usage

```
.mcee_message_if_no_availability_provided(availability, verbose)
```

Arguments

availability	Availability column name (or NULL)
verbose	Logical; whether to print messages

Value

Invisibly TRUE

.mcee_print_coef_table

Print formatted coefficient table for MCEE results

Description

Print formatted coefficient table for MCEE results

Usage

```
.mcee_print_coef_table(tab)
```

Arguments

tab	Data frame with coefficient estimates, standard errors, etc.

Value

Used for side effects (printing)

.mcee_require_cols *Check that required columns exist in data frame*

Description

Check that required columns exist in data frame

Usage

```
.mcee_require_cols(data, cols, where = "data")
```

Arguments

data	Data frame to check
cols	Character vector of required column names
where	Context description for error messages

Value

Invisibly TRUE if all columns exist, otherwise stops with error

.mcee_resolve_rand_prob *Resolve randomization probability from column name or scalar*

Description

Resolve randomization probability from column name or scalar

Usage

```
.mcee_resolve_rand_prob(data, rand_prob, availability = NULL)
```

Arguments

data	Data frame
rand_prob	Either column name or numeric scalar
availability	Optional availability column name for validation

Value

Numeric vector of randomization probabilities

.mcee_validate_clipping *Validate clipping bounds for probability predictions*

Description

Validate clipping bounds for probability predictions

Usage

```
.mcee_validate_clipping(clipping)
```

Arguments

clipping	Numeric vector of length 2 with lower and upper bounds

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_validate_method *Validate that learning method is supported*

Description

Validate that learning method is supported

Usage

```
.mcee_validate_method(method)
```

Arguments

method	Method name to validate

Value

Invisibly TRUE if valid, otherwise stops with error

.mcee_vars_in_config *Extract variables from nuisance configuration formula*

Description

Extract variables from nuisance configuration formula

Usage

```
.mcee_vars_in_config(cfg)
```

Arguments

cfg	Configuration list (may contain formula element)

Value

Character vector of variable names from config formula

.mcee_vars_in_rhs *Extract variable names from RHS-only formula*

Description

Extract variable names from RHS-only formula

Usage

```
.mcee_vars_in_rhs(rhs_only_formula)
```

Arguments

rhs_only_formula	RHS-only formula object

Value

Character vector of variable names (empty if not a valid formula)

`data_binary` *A synthetic data set of an MRT with binary proximal outcomes*

Description

Baseline model:

$$\log E\{Y_{t+1} \mid A_t = 0, I_t = 1\} = \alpha_0 + \alpha_1 \cdot \text{time/total\_T} + \alpha_2 \cdot \mathbf{1}(\text{time} > \text{total\_T}/2).$$

Treatment effect model:

$$\log RR_t = \beta_0 + \beta_1 \cdot \text{time/total\_T}.$$

Randomization probabilities p_t cycle over 0.3, 0.5, 0.7 (with repetition). Availability is exogenous at 0.8 for all time points.

Usage

```
data_binary
```

Format

A data frame with 3000 observations and 10 variables:

userid Individual id number.

time Decision point index.

time_var1 Time-varying covariate 1, the \"standardized time in study\", defined as the current decision point index divided by the total number of decision points.

time_var2 Time-varying covariate 2, indicator of \"the second half of the study\", defined as whether the current decision point index is greater than the total number of decision points divided by 2.

Y Binary proximal outcome.

A Treatment assignment: whether the intervention is randomized to be delivered (=1) or not (=0) at the current decision point.

rand_prob Randomization probability $P(A = 1)$ for the current decision point.

avail Availability indicator (=1 available, =0 not available) at the current decision point.

data_distal_continuous

A synthetic data set of an MRT with continuous distal outcome

Description

Simulated longitudinal dataset suitable for illustrating the ‘dcee()‘ function. Each row corresponds to one decision point for one subject. The distal outcome ‘Y‘ is constant within subject (because it is measured at the end of the study, and here we append it to the long format data as an extra column to conform with the ‘dcee()‘ function requirement.

Usage

```
data_distal_continuous
```

Format

a data frame with 1500 observations and 11 variables

userid Subject identifier

dp Decision point (1..T)

X Endogenous continuous time-varying covariate

Z Endogenous binary time-varying covariate

avail Availability indicator (0/1)

A Treatment (0/1)

prob_A Randomization probability P(A=1|H_t)

A_lag1 Lagged treatment

Y Distal continuous outcome (constant per subject)

data_mimicHeartSteps

A synthetic data set that mimics the HeartSteps V1 data structure to illustrate the use of [wcls()] function for continuous proximal outcomes

Description

A synthetic data set that mimics the HeartSteps V1 data structure to illustrate the use of [wcls()] function for continuous proximal outcomes

Usage

```
data_mimicHeartSteps
```

Format

a data frame with 7770 observations and 9 variables

userid individual id number

time decision point index

day_in_study day in the study

logstep_30min proximal outcome: the step count in the 30 minutes following the current decision point (log-transformed)

logstep_30min_lag1 proximal outcome at the previous decision point (lag-1 outcome): the step count in the 30 minutes following the previous decision point (log-transformed)

logstep_pre30min the step count in the 30 minutes prior to the current decision point (log-transformed); used as a control variable

is_at_home_or_work whether the individual is at home or work (=1) or at other locations (=0) at the current decision point

intervention whether the intervention is randomized to be delivered (=1) or not (=0) at the current decision point

rand_prob the randomization probability P(A=1) for the current decision point

availability whether the individual is available (=1) or not (=0) at the current decision point

---

data_time_varying_mediator_distal_outcome

Example longitudinal dataset with time-varying mediator and distal outcome

---

Description

A simulated long-format dataset used in the vignette and tests. Each row corresponds to one subject–decision point. The distal outcome 'Y' is constant within subject (repeated on every row for that subject).

Usage

```
data_time_varying_mediator_distal_outcome
```

Format

A data frame with `n * T_val` rows and the following columns:

id Subject identifier (integer).

dp Decision point index, strictly increasing within subject (integer).

I Availability indicator at time `dp` (0/1).

A Treatment at time `dp` (0/1).

M Mediator at time `dp` (numeric; could be binary or continuous).

X Time-varying covariate at time dp (numeric).

A_prev Lagged treatment at time dp-1 (0/1).

M_prev Lagged mediator at time dp-1 (numeric).

X_prev Lagged covariate at time dp-1 (numeric).

I_prev Lagged availability at time dp-1 (0/1).

p_A Randomization probability for A at time dp (numeric in (0,1)).

p_I Availability probability for I at time dp (numeric in (0,1)).

mu_M Conditional mean of M given history (numeric; from DGM).

mu_X Conditional mean of X given history (numeric; from DGM).

mu_Y Conditional mean component for distal outcome Y (numeric; from DGM).

Y Distal outcome, constant within subject (numeric).

Details

Generated by dgm_time_varying_mediator_distal_outcome() in the package source. Intended for illustrating mcee usage. No missing values.

Source

Simulated.

See Also

mcee, mcee_general, mcee_userfit_nuisance

Examples

```
data(data_time_varying_mediator_distal_outcome)
str(data_time_varying_mediator_distal_outcome)
```

---

dcee *Distal Causal Excursion Effect (DCEE) Estimation*

Description

Fits distal causal excursion effects in micro-randomized trials using a **two-stage** estimator: (i) learn nuisance outcome regressions $\mu_a(H_t)$ with a specified learner (parametric/ML), optionally with cross-fitting; (ii) solve estimating equations for the distal excursion effect parameters (β).

This wrapper standardizes inputs and delegates computation to [dcee_helper_2stage_estimation()].

Usage

```
dcee(
  data,
  id,
  outcome,
  treatment,
  rand_prob,
  moderator_formula,
  control_formula,
  availability = NULL,
 control_reg_method = c("gam", "lm", "rf", "ranger", "sl", "sl.user-specified-library",
    "set_to_zero"),
  cross_fit = FALSE,
  cf_fold = 10,
  weighting_function = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

data	A data.frame in long format.
id	Character scalar: column name for subject identifier.
outcome	Character scalar: column name for proximal/distal outcome.
treatment	Character scalar: column name for binary treatment {0,1}.
rand_prob	Character scalar: column name for randomization probability giving $P(A_t = 1 \mid H_t)$ (must lie in (0,1)).
moderator_formula	RHS-only formula of moderators of the excursion effect (e.g., '~ 1', '~ Z', or '~ Z1 + Z2').
control_formula	RHS-only formula of covariates for learning nuisance outcome regressions. When 'control_reg_method = "gam"', 's(x)' terms are allowed (e.g., '~ x1 + s(x2)'). For SuperLearner methods, variables are extracted from this formula to build the design matrix 'X'.
availability	Optional character scalar: column name for availability indicator (0/1). If 'NULL', availability is taken as 1 for all rows.
control_reg_method	One of '"gam"', '"lm"', '"rf"', '"ranger"', '"sl"', '"sl.user-specified-library"', '"set_to_zero"'. See Details.
cross_fit	Logical; if 'TRUE', perform K-fold cross-fitting by subject id.
cf_fold	Integer; number of folds if 'cross_fit = TRUE' (default 10).
weighting_function	Either a single numeric constant applied to all rows, or a character column name in 'data' giving decision-point weights ω_t.

verbose	Logical; print minimal preprocessing messages (default 'TRUE').
...	Additional arguments passed through to the chosen learner (e.g., 'num.trees', 'mtry' for random forests; 'sl.library' when 'control_reg_method = "sl.user-specified-library"').

Details

Learners. - 'gam' uses **mgcv** and supports 's(.)' terms in 'control_formula'. - 'lm' uses base `stats::lm`. - 'rf' uses **randomForest**; 'ranger' uses **ranger**. - 'sl' / 'sl.user-specified-library' use **SuperLearner**. For the former, 'sl.library = c("SL.mean", "SL.glm", "SL.earth")' are used. For the latter, please provide 'sl.library = c("SL.mean", ...)' via '...'.

Notes. - Treatment must be coded 0/1; 'rand_prob' must lie strictly in (0,1). - 'control_formula = ~ 1' is only valid with 'control_reg_method = "set_to_zero"'.

Value

An object of class '"dcee_fit"' with components:

call The matched call to `dcee()`.

fit A list returned by the two–stage helper with elements:

- `beta_hat` Named numeric vector of distal causal excursion effect estimates β. Names are `"Intercept"` and the moderator names (if any) from `moderator_formula`.
- `beta_se` Named numeric vector of standard errors for `beta_hat` (same order/names).
- `beta_varcov` Variance–covariance matrix of `beta_hat` (square matrix; row/column names match `names(beta_hat)`).
- `conf_int` Matrix of large-sample (normal) Wald 95% confidence intervals for `beta_hat`; columns are `"2.5 %"` and `"97.5 %"`.
- `conf_int_tquantile` Matrix of small-sample (t-quantile) 95% confidence intervals for `beta_hat`; columns are `"2.5 %"` and `"97.5 %"`; degrees of freedom are provided in `$df` of the `"dcee_fit"` object.
- `regfit_a0` Stage-1 nuisance regression fit for $\mu_0(H_t)$ (outcome model among `A=0`), or `NULL` when `control_reg_method = "set_to_zero"`. **Note:** when `cross_fit = TRUE`, this is the learner object from the *last* fold and is provided for inspection only (do not use for out-of-fold prediction).
- `regfit_a1` Stage-1 nuisance regression fit for $\mu_1(H_t)$ (outcome model among `A=1`); same caveats as `regfit_a0` regarding `cross_fit`.

df Small-sample degrees of freedom used for t-based intervals: number of unique subjects minus `length(fit$beta_hat)`.

Examples

```
data(data_distal_continuous, package = "MRTAnalysis")

## Fast example: marginal effect with linear nuisance (CRAN-friendly)
fit_lm <- dcee(
    data = data_distal_continuous,
    id = "userid", outcome = "Y", treatment = "A", rand_prob = "prob_A",
    moderator_formula = ~1, # marginal (no moderators)
```

```
    control_formula = ~X, # simple linear nuisance
    availability = "avail",
    control_reg_method = "lm",
    cross_fit = FALSE
)
summary(fit_lm)
summary(fit_lm, show_control_fit = TRUE) # show Stage-1 fit info

## Moderated effect with GAM nuisance (allows smooth terms); may be slower

fit_gam <- dcee(
    data = data_distal_continuous,
    id = "userid", outcome = "Y", treatment = "A", rand_prob = "prob_A",
    moderator_formula = ~Z, # test moderation by Z
    control_formula = ~ s(X) + Z, # smooth in nuisance via mgcv::gam
    availability = "avail",
    control_reg_method = "gam",
    cross_fit = TRUE, cf_fold = 5
)
summary(fit_gam, lincomb = c(0, 1)) # linear combo: the Z coefficient
summary(fit_gam, show_control_fit = TRUE) # show Stage-1 fit info


## Optional: SuperLearner (runs only if installed)

library(SuperLearner)
fit_sl <- dcee(
    data = data_distal_continuous,
    id = "userid", outcome = "Y", treatment = "A", rand_prob = "prob_A",
    moderator_formula = ~1,
    control_formula = ~ X + Z,
    availability = "avail",
    control_reg_method = "sl",
    cross_fit = FALSE
)
summary(fit_sl)
```

---

emee *Estimates the causal excursion effect for binary outcome MRT*

Description

Returns the estimated causal excursion effect (on log relative risk scale) and the estimated standard error. Small sample correction using the "Hat" matrix in the variance estimate is implemented.

Usage

```
emee(
  data,
```

```
  id,
  outcome,
  treatment,
  rand_prob,
  moderator_formula,
  control_formula,
  availability = NULL,
  numerator_prob = NULL,
  start = NULL,
  verbose = TRUE
)
```

Arguments

`data`	A data set in long format.
`id`	The subject id variable.
`outcome`	The outcome variable.
`treatment`	The binary treatment assignment variable.
`rand_prob`	The randomization probability variable.
`moderator_formula`	A formula for the moderator variables. This should start with ~ followed by the moderator variables. When set to `~ 1`, a fully marginal excursion effect (no moderators) is estimated.
`control_formula`	A formula for the control variables. This should start with ~ followed by the control variables. When set to `~ 1`, only an intercept is included as the control variable.
`availability`	The availability variable. Use the default value (`NULL`) if your MRT doesn't have availability considerations.
`numerator_prob`	Either a number between 0 and 1, or a variable name for a column in data. If you are not sure what this is, use the default value (`NULL`).
`start`	A vector of the initial value of the estimators used in the numerical solver. If using default value (`NULL`), a vector of 0 will be used internally. If specifying a non-default value, this needs to be a numeric vector of length (number of moderator variables including the intercept) + (number of control variables including the intercept).
`verbose`	If default ('TRUE'), additional messages will be printed during data preprocessing.

Value

An object of type "emee_fit"

Examples

```
## estimating the fully marginal excursion effect by setting
## moderator_formula = ~ 1
emee(
    data = data_binary,
    id = "userid",
    outcome = "Y",
    treatment = "A",
    rand_prob = "rand_prob",
    moderator_formula = ~1,
    control_formula = ~ time_var1 + time_var2,
    availability = "avail"
)

## estimating the causal excursion effect moderated by time_var1
## by setting moderator_formula = ~ time_var1
emee(
    data = data_binary,
    id = "userid",
    outcome = "Y",
    treatment = "A",
    rand_prob = "rand_prob",
    moderator_formula = ~time_var1,
    control_formula = ~ time_var1 + time_var2,
    availability = "avail"
)
```

---

emee2 *Estimates the causal excursion effect for binary outcome MRT*

Description

Returns the estimated causal excursion effect (on log relative risk scale) and the estimated standard error. Small sample correction using the "Hat" matrix in the variance estimate is implemented. This is a slightly altered version of `emee()`, where the treatment assignment indicator is also centered in the residual term. It would have similar (but not exactly the same) numerical output as `emee()`. This is the estimator based on which the sample size calculator for binary outcome MRT is developed. (See R package `MRTSampleSizeBinary`.)

Usage

```
emee2(
  data,
  id,
  outcome,
  treatment,
  rand_prob,
  moderator_formula,
```

```
  control_formula,
  availability = NULL,
  numerator_prob = NULL,
  start = NULL,
  verbose = TRUE
)
```

Arguments

data	A data set in long format.
id	The subject id variable.
outcome	The outcome variable.
treatment	The binary treatment assignment variable.
rand_prob	The randomization probability variable.
moderator_formula	A formula for the moderator variables. This should start with ~ followed by the moderator variables. When set to ~ 1, a fully marginal excursion effect (no moderators) is estimated.
control_formula	A formula for the control variables. This should start with ~ followed by the control variables. When set to ~ 1, only an intercept is included as the control variable.
availability	The availability variable. Use the default value (NULL) if your MRT doesn't have availability considerations.
numerator_prob	Either a number between 0 and 1, or a variable name for a column in data. If you are not sure what this is, use the default value (NULL).
start	A vector of the initial value of the estimators used in the numerical solver. If using default value (NULL), a vector of 0 will be used internally. If specifying a non-default value, this needs to be a numeric vector of length (number of moderator variables including the intercept) + (number of control variables including the intercept).
verbose	If default ('TRUE'), additional messages will be printed during data preprocessing.

Value

An object of type "emee_fit"

Examples

```
## estimating the fully marginal excursion effect by setting
## moderator_formula = ~ 1
emee2(
    data = data_binary,
    id = "userid",
    outcome = "Y",
    treatment = "A",
```

```
    rand_prob = "rand_prob",
    moderator_formula = ~1,
    control_formula = ~ time_var1 + time_var2,
    availability = "avail"
)

## estimating the causal excursion effect moderated by time_var1
## by setting moderator_formula = ~ time_var1
emee2(
    data = data_binary,
    id = "userid",
    outcome = "Y",
    treatment = "A",
    rand_prob = "rand_prob",
    moderator_formula = ~time_var1,
    control_formula = ~ time_var1 + time_var2,
    availability = "avail"
)
```

---

mcee *Mediated Causal Excursion Effects for MRTs (streamlined)*

Description

Estimates the Natural Direct Excursion Effect (NDEE; α) and Natural Indirect Excursion Effect (NIEE; β) for distal outcomes in micro-randomized trials (MRTs). Assumes the randomization probability is known (via `rand_prob`) and fits all nuisance functions using the same learner specified by `control_reg_method`.

Usage

```
mcee(
  data,
  id,
  dp,
  outcome,
  treatment,
  mediator,
  availability = NULL,
  rand_prob,
  time_varying_effect_form,
  control_formula_with_mediator,
  control_reg_method = c("glm", "gam", "rf", "ranger", "sl"),
  weight_per_row = NULL,
  specific_dp_only = NULL,
  verbose = TRUE,
  SL.library = NULL
)
```

Arguments

`data`	A data.frame in long format (one row per id-by-decision point).
`id`	Character. Column name for subject identifier.
`dp`	Character. Column name for decision point index (must increase strictly within subject).
`outcome`	Character. Column name for distal outcome (constant within subject).
`treatment`	Character. Column name for treatment (coded 0/1).
`mediator`	Character. Column name for mediator.
`availability`	Optional character. Column name for availability (0/1). If `NULL`, all rows are treated as available.
`rand_prob`	Either a column name in `data` or a scalar giving the known randomization probability $P(A_t = 1 \mid H_t)$. (Technically, this is $P(A_t = I_t \mid H_t)$, but the user is allowed to input $P(A_t = 1 \mid H_t)$ and the function will automatically correct it by setting `p1 = 1` when $I_t = 0$.)
`time_varying_effect_form`	RHS-only formula for the basis $f(t)$ (e.g., `~ 1`, `~ dp`, `~ poly(dp,2)`).
`control_formula_with_mediator`	RHS-only formula for control variables used in nuisance models that may include the mediator (the wrapper will drop the mediator internally for nuisances that must exclude it).
`control_reg_method`	Learner for nuisance fits: one of `"glm"`, `"gam"`, `"rf"`, `"ranger"`, `"sl"`.
`weight_per_row`	Optional numeric vector of row weights (nonnegative, length `nrow(data)`). If `NULL`, uniform within-id weights are used.
`specific_dp_only`	Optional numeric vector of decision points to target; internally converted to `weight_per_row`.
`verbose`	Logical; print progress messages.
`SL.library`	Optional character vector of SuperLearner libraries (used when `control_reg_method = "sl"`).

Details

Requirements: rows grouped by subject, strictly increasing `dp` within subject, no missing (`NA`/`NaN`/`Inf`) in relevant variables. If `availability` is supplied, the wrapper enforces at $I = 0$: $p_1 = q_1 = 1$ in the nuisances.

Value

An object of class `"mcee_fit"` with elements:

- `mcee_fit`: list with `alpha_hat`, `beta_hat`, `alpha_se`, `beta_se`, `varcov`, `alpha_varcov`, `beta_varcov`.
- `nuisance_models`: fitted Stage-1 models for `p,q,eta,mu,nu`.

- nuisance_fitted: per-row fitted values for the nuisance functions.
- meta: list with basis dimension, number of ids, per-id lengths, weights used.
- call: the matched call.

See Also

summary.mcee_fit, mcee_general, mcee_userfit_nuisance

Examples

```
set.seed(1)
n <- 10
T <- 4
id <- rep(1:n, each = T)
dp <- rep(1:T, times = n)
A <- rbinom(n * T, 1, 0.5)
M <- rbinom(n * T, 1, plogis(-0.2 + 0.3 * A + 0.1 * dp))
Y <- ave(0.5 * A + 0.6 * M + 0.1 * dp + rnorm(n * T), id)
dat <- data.frame(id, dp, A, M, Y)

fit <- mcee(dat, "id", "dp", "Y", "A", "M",
    time_varying_effect_form = ~1,
    control_formula_with_mediator = ~ dp + M,
    control_reg_method = "glm",
    rand_prob = 0.5, verbose = TRUE
)
summary(fit)
```

---

mcee_config_gam *Configure GAM for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using generalized additive models via mgcv::gam(). Supports smooth terms like s().

Usage

```
mcee_config_gam(target, formula, family = NULL, clipping = NULL)
```

Arguments

target	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
formula	RHS-only formula (e.g., ~ X1 + s(time) + s(X2, k=5)).
family	Optional GLM family. Defaults to binomial() for "p"/"q", gaussian() for "eta"/"mu"/"nu".
clipping	Optional numeric vector c(lo, hi) to clip predictions into [lo, hi] for numerical stability.

Value

A configuration list for use with mcee_general.

Examples

```
# GAM with smooth time effect
cfg_eta <- mcee_config_gam("eta", ~ X1 + s(dp, k = 4))

# GAM with multiple smooths
cfg_mu <- mcee_config_gam("mu", ~ s(dp) + s(M, X1, k = 10))
```

---

mcee_config_glm *Configure GLM for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using generalized linear models via `stats::glm()`.

Usage

```
mcee_config_glm(target, formula, family = NULL, clipping = NULL)
```

Arguments

target	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
formula	RHS-only formula (e.g., ~ X1 + X2 + poly(time, 2)).
family	Optional GLM family. Defaults to binomial() for "p"/"q", gaussian() for "eta"/"mu"/"nu".
clipping	Optional numeric vector c(lo, hi) to clip predictions into [lo, hi] for numerical stability.

Value

A configuration list for use with mcee_general.

Examples

```
# Binary outcome model for propensity
cfg_q <- mcee_config_glm("q", ~ dp + M, family = binomial())

# Gaussian outcome model
cfg_eta <- mcee_config_glm("eta", ~ dp + X1)
```

mcee_config_known *Configure known constant values for MCEE nuisance parameters*

Description

Creates a configuration for nuisance parameters with known constant values, bypassing model fitting. Useful for known randomization probabilities in MRTs.

Usage

```
mcee_config_known(target, value = NULL, a1 = NULL, a0 = NULL)
```

Arguments

target	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
value	Numeric scalar. Single constant value for all observations.
a1, a0	Numeric scalars. Arm-specific constants for A=1 and A=0 conditions. If provided, these override value.

Value

A configuration list for use with mcee_general.

Examples

```
# Known randomization probability
cfg_p <- mcee_config_known("p", 0.6)

# Arm-specific known values
cfg_eta <- mcee_config_known("eta", a1 = 0.8, a0 = 0.2)
```

mcee_config_lm *Configure linear model for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using linear models via stats::lm(). Only appropriate for continuous outcomes.

Usage

```
mcee_config_lm(target, formula)
```

Arguments

target	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
formula	RHS-only formula (e.g., ~ X1 + X2 + poly(dp, 2)).

Value

A configuration list for use with mcee_general.

Examples

```
# Linear model for continuous outcome
cfg_eta <- mcee_config_lm("eta", ~ dp + X1 + X2)
```

`mcee_config_maker` *Build a nuisance-configuration object for* `mcee_general()`

Description

Creates a configuration list describing **how to obtain a nuisance function** used by mcee_general. You may either:

1. supply **known values** (bypasses learning), or
2. specify a **learning method** (e.g., GLM/GAM/RF/Ranger/SL) with a formula.

Usage

```
mcee_config_maker(
  target,
  method = NULL,
  formula = NULL,
  family = NULL,
  known = NULL,
  known_a1 = NULL,
  known_a0 = NULL,
  clipping = NULL,
  SL.library = NULL,
  ...
)
```

Arguments

`target`	Character; which nuisance to configure. One of `"p"`, `"q"`, `"eta"`, `"mu"`, `"nu"`.
`method`	Optional character learner name when *not* using known values. Supported: • `"glm"`, `"gam"`, `"lm"` (formula-based); • `"rf"` (randomForest), `"ranger"`; • `"sl"` (SuperLearner). Ignored if any of `known`, `known_a1`, `known_a0` is provided.
`formula`	RHS-only formula describing predictors for the learner (used when `method` is formula-based; ignored for `known`). For `method = "gam"`, `s()` terms are allowed (via **mgcv**).

family	Optional GLM/GAM family. If NULL, a default is chosen based on target and method (typically binomial() for p/q, gaussian() for eta/mu/nu).
known	Optional numeric scalar/vector of **known values** for the nuisance. Commonly used for target = "p" in MRTs when the randomization probability is known.
known_a1, known_a0	Optional numeric scalar/vector providing known values for the *treatment-specific* versions of a nuisance (e.g., eta1/eta0, mu1/mu0, nu1/nu0). If supplied, the function returns a "known" config and method/formula/family are ignored.
clipping	Optional numeric vector of length 2, c(lower, upper), to truncate predictions (e.g., probabilities into $[\epsilon, 1-\epsilon]$). Validated but not required.
SL.library	Character vector of SuperLearner libraries (only used when method = "sl"). If NULL, defaults to c("SL.mean", "SL.glm", "SL.gam").
...	Reserved for future extensions; currently ignored.

Details

If any of known, known_a1, or known_a0 is provided, the returned configuration is of type "known" and **no learner will be fit**. Otherwise, the configuration records the requested learner, formula, family, optional clipping, and (for SL) the library.

Internally, helper validators ensure method is supported and clipping (if provided) is sane. Family defaults are chosen when family = NULL for GLM/GAM methods.

Value

A named list describing the configuration. For known configs:

```
list(
  nuisance_parameter = <target>,
  known   = <numeric or NULL>,
  known_a1 = <numeric or NULL>,
  known_a0 = <numeric or NULL>,
  clipping = <numeric length-2 or NULL>
)
```

For learner configs:

```
list(
  nuisance_parameter = <target>,
  method  = <character>,
  formula = <formula or NULL>,
  family  = <family or NULL>,
  clipping = <numeric length-2 or NULL>,
  SL.library = <character vector; only when method == "sl">
)
```

See Also

mcee_general, helper constructors like mcee_config_known(), mcee_config_glm(), mcee_config_gam(), mcee_config_lm(), mcee_config_rf(), mcee_config_ranger(), mcee_config_sl(), mcee_config_sl_user().

Examples

```
# Known p (MRT randomization), GLM for other nuisances
cfg_p <- mcee_config_maker("p", known = 0.5)
cfg_q <- mcee_config_maker("q", method = "glm", formula = ~ dp + M)
cfg_eta <- mcee_config_maker("eta", method = "glm", formula = ~dp)
cfg_mu <- mcee_config_maker("mu", method = "glm", formula = ~ dp + M)
cfg_nu <- mcee_config_maker("nu", method = "glm", formula = ~dp)

# SuperLearner with default library (set explicitly if you prefer)
# cfg_q_sl <- mcee_config_maker("q", method = "sl", formula = ~ dp + M,
#                               SL.library = c("SL.mean","SL.glm","SL.ranger"))

# Known treatment-specific outcome regressions (e.g., from external source)
# cfg_eta_known <- mcee_config_maker("eta", known_a1 = rep(1, 100),
#                                         known_a0 = rep(0, 100))
```

---

mcee_config_ranger *Configure Ranger Random Forest for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using ranger random forests via ranger::ranger(). Faster alternative to randomForest.

Usage

```
mcee_config_ranger(target, formula)
```

Arguments

target	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
formula	RHS-only formula (e.g., ~ X1 + X2 + dp).

Value

A configuration list for use with mcee_general.

Examples

```
# Ranger random forest for outcome model
cfg_eta <- mcee_config_ranger("eta", ~ dp + X1 + X2 + X3)
```

mcee_config_rf *Configure Random Forest for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using random forests via `randomForest::randomForest()`. Good for nonlinear patterns.

Usage

```
mcee_config_rf(target, formula)
```

Arguments

`target`	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
`formula`	RHS-only formula (e.g., `~ X1 + X2 + dp`).

Value

A configuration list for use with `mcee_general`.

Examples

```
# Random forest for complex propensity model
cfg_q <- mcee_config_rf("q", ~ dp + M + X1 + X2)
```

mcee_config_sl *Configure SuperLearner for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using SuperLearner via `SuperLearner::SuperLearner()`. Automatically selects among multiple learning algorithms.

Usage

```
mcee_config_sl(target, formula, SL.library = NULL, clipping = NULL)
```

Arguments

`target`	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
`formula`	RHS-only formula (e.g., `~ X1 + X2 + dp`).
`SL.library`	Optional character vector of learner names. If `NULL`, uses default library: `c("SL.mean", "SL.glm", "SL.gam")`.
`clipping`	Optional numeric vector `c(lo, hi)` to clip predictions into [lo, hi] for numerical stability.

Value

A configuration list for use with mcee_general.

Examples

```
# SuperLearner with default library
cfg_q <- mcee_config_sl("q", ~ dp + M + X1)

# SuperLearner with custom library
cfg_eta <- mcee_config_sl("eta", ~ dp + X1,
    SL.library = c("SL.glm", "SL.rf", "SL.ranger")
)
```

---

mcee_config_sl_user *Configure SuperLearner with user-specified library for MCEE nuisance parameters*

Description

Creates a configuration to fit nuisance parameters using SuperLearner with a user-specified library (required parameter).

Usage

```
mcee_config_sl_user(target, formula, SL.library, clipping = NULL)
```

Arguments

target	Character. Nuisance parameter name ("p", "q", "eta", "mu", "nu").
formula	RHS-only formula (e.g., ~ X1 + X2 + dp).
SL.library	Character vector of learner names (required).
clipping	Optional numeric vector c(lo, hi) to clip predictions into [lo, hi] for numerical stability.

Value

A configuration list for use with mcee_general.

Examples

```
# SuperLearner with specific library
cfg_mu <- mcee_config_sl_user("mu", ~ dp + M + X1,
    SL.library = c("SL.glm", "SL.earth", "SL.nnet")
)
```

mcee_general *Mediated Causal Excursion Effects (configurable nuisance models)*

Description

Like mcee, but each nuisance function is configured explicitly via config_* objects (formula/method/family or known).

Usage

```
mcee_general(
  data,
  id,
  dp,
  outcome,
  treatment,
  mediator,
  availability = NULL,
  time_varying_effect_form,
  config_p,
  config_q,
  config_eta,
  config_mu,
  config_nu,
  weight_per_row = NULL,
  verbose = TRUE
)
```

Arguments

data	A data.frame in long format (one row per id-by-decision point).
id	Character. Column name for subject identifier.
dp	Character. Column name for decision point index (must increase strictly within subject).
outcome	Character. Column name for distal outcome (constant within subject).
treatment	Character. Column name for treatment (coded 0/1).
mediator	Character. Column name for mediator.
availability	Optional character. Column name for availability (0/1). If NULL, all rows are treated as available.
time_varying_effect_form	RHS-only formula for the basis $f(t)$ (e.g., ~ 1, ~ dp, ~ poly(dp,2)).
config_p, config_q, config_eta, config_mu, config_nu	Lists created by mcee_config_maker() or helpers (e.g., mcee_config_glm(), mcee_config_known(), mcee_config_ranger(), etc.).

weight_per_row	Optional numeric vector of row weights (nonnegative, length nrow(data)). If NULL, uniform within-id weights are used.
verbose	Logical; print progress messages.

Details

Use this wrapper for observational studies (estimate p) or when you want different learners per nuisance. The same data requirements as mcee apply.

Value

An "mcee_fit" object; see mcee.

See Also

mcee, mcee_userfit_nuisance, mcee_config_maker

Examples

```
set.seed(1)
n <- 10
T <- 4
id <- rep(1:n, each = T)
dp <- rep(1:T, times = n)
A <- rbinom(n * T, 1, 0.5)
M <- rbinom(n * T, 1, plogis(-0.2 + 0.3 * A + 0.1 * dp))
Y <- ave(0.5 * A + 0.6 * M + 0.1 * dp + rnorm(n * T), id)
dat <- data.frame(id, dp, A, M, Y)

cfg <- list(
    p   = mcee_config_known("p", 0.5),
    q   = mcee_config_glm("q", ~ dp + M),
    eta = mcee_config_glm("eta", ~dp),
    mu  = mcee_config_glm("mu", ~ dp + M),
    nu  = mcee_config_glm("nu", ~dp)
)
fit_gen <- mcee_general(dat, "id","dp","Y","A","M",
    time_varying_effect_form = ~ dp,
   config_p=cfg$p, config_q=cfg$q, config_eta=cfg$eta, config_mu=cfg$mu, config_nu=cfg$nu)
```

---

mcee_helper_2stage_estimation

Two-stage helper for mediated causal excursion effects (MCEE)

Description

Fits all nuisance components (Stage 1) and then computes the MCEE parameters (Stage 2) and their sandwich variance. This is a low-level driver used by the high-level wrapper; it assumes ‘omega_nrows‘ and ‘f_nrows‘ are already aligned to the rows of ‘data‘.

Usage

```
mcee_helper_2stage_estimation(
  data,
  id_var,
  dp_var,
  outcome_var,
  treatment_var,
  mediator_var,
  avail_var = NULL,
  config_p,
  config_q,
  config_eta,
  config_mu,
  config_nu,
  omega_nrows,
  f_nrows
)
```

Arguments

data	A long-format 'data.frame' (one row per subject-by-decision point).
id_var	Character scalar. Name of the subject ID column.
dp_var	Character scalar. Name of the decision point column (values need not be consecutive; they may vary in count across subjects).
outcome_var	Character scalar. Name of the distal outcome column.
treatment_var	Character scalar. Name of the binary treatment column (coded 0/1).
mediator_var	Character scalar. Name of the mediator column.
avail_var	Character scalar or 'NULL'. Name of the availability column (1 = available, 0 = unavailable). If 'NULL', availability is treated as all 1.
config_p	Configuration for $p_t(a \mid H_t)$ (propensity). A **list** using one of the following schemas: • *Known constant(s)* (skips fitting): `list(known = c(...))` or arm-specific `known_a1`/`known_a0`. • *Model fit*: `list(formula = ~ rhs, method = m, ...)` where `method` is one of `"glm"`, `"gam"`, `"rf"`, `"ranger"`, `"sl"`, `"sl.user-specified-library"`. Optional fields: – `family`: a GLM/GAM family. If omitted, **auto-detected** as `binomial()` for p and q, otherwise `gaussian()`. – `clipping`: numeric length-2 `c(lo, hi)` to clip probabilities into `[lo, hi]` (useful for stability). – For `method = "sl"`: `SL.library` (character vector of learners); if omitted, a simple default library is used: c("SL.mean", "SL.glm", "SL.gam").
config_q	Configuration for $q_t(a \mid H_t, M_t)$. Same schema as `config_p`.
config_eta	Configuration for $\eta_t(a, H_t)$ (outcome given A, H). Same schema as `config_p`; default family auto-detected to `gaussian()` if omitted.

`config_mu` Configuration for $\mu_t(a, H_t, M_t)$ (outcome given A, H, M). Same schema as `config_p`; default family auto-detected to `gaussian()` if omitted.

`config_nu` Configuration for $\nu_t(a, H_t)$ (cross-world ICE based on μ). Same schema as `config_p`; default family auto-detected to `gaussian()` if omitted.

`omega_nrows` Numeric vector of length `nrow(data)`. Per-row weights $\omega(i,t) \geq 0$. Rows are aligned with `data` (no reordering).

`f_nrows` Numeric matrix with `nrow(data)` rows and `p` columns. Row `r` contains $f(t_r)^\top$ (the basis evaluated at the decision point of row `r`). The same basis is used for both α (NDEE) and β (NIEE).

Details

Availability handling: When `avail_var` exists and equals 0, Stage 1 sets the working probabilities to 1 for that row (e.g., $\hat{p}_t(1 \mid H_t) = 1$, $\hat{p}_t(0 \mid H_t) = 1$, similarly for $\hat{q}_t$). This prevents division-by-zero in the estimating equations.

Auto-family rules: If `family` is omitted in a GLM/GAM config, it defaults to `binomial()` for `config_p` and `config_q`, and to `gaussian()` for `config_eta`, `config_mu`, and `config_nu`.

Learners:

- `"glm"`: uses `stats::glm()`.
- `"gam"`: uses `mgcv::gam()` (supports `s()` smooths).
- `"rf"`: uses `randomForest::randomForest()`.
- `"ranger"`: uses `ranger::ranger()`.
- `"sl"`: uses `SuperLearner::SuperLearner()`. If `SL.library` is not given, a simple default library is used: c("SL.mean", "SL.glm", "SL.gam").

Value

A list with components:

`fit` A list with entries `alpha_hat`, `alpha_se`, `beta_hat`, `beta_se`, and `varcov` (the $2p \times 2p$ sandwich variance for $(\alpha^\top, \beta^\top)^\top$).

`nuisance_models` A list of fitted Stage-1 objects: `p`, `q`, `eta1`, `eta0`, `mu1`, `mu0`, `nu1`, `nu0`. (For `known`/`known_a0`/`known_a1`, a small descriptor list is returned.)

See Also

`mcee_general` for a high-level wrapper that constructs `omega_nrows` and `f_nrows` from user-friendly arguments.

Examples

```
## Not run:
# Minimal sketch (assuming `df` has columns id, t, A, M, Y, I):
fit <- mcee_helper_2stage_estimation(
    data = df,
    id_var = "id", dp_var = "t", outcome_var = "Y",
```

```
    treatment_var = "A", mediator_var = "M", avail_var = "I",
    config_p = list(formula = ~ t + M, method = "glm"), # binomial auto
    config_q = list(formula = ~ t + M + A, method = "glm"), # binomial auto
    config_eta = list(formula = ~t, method = "gam"), # gaussian auto
    config_mu = list(formula = ~ t + s(M), method = "gam"), # gaussian auto
    config_nu = list(formula = ~t, method = "glm"), # gaussian auto
    omega_nrows = rep(1, nrow(df)),
    f_nrows = cbind(1) # marginal (p = 1)
)
fit$fit$alpha_hat
fit$fit$beta_hat

## End(Not run)
```

`mcee_helper_stage1_fit_nuisance` *Fit all nuisance models for MCEE Stage 1*

Description

Fits all five nuisance components required for MCEE estimation and returns both per-row predictions and fitted model objects. This is Stage 1 of the two-stage MCEE procedure.

Usage

```
mcee_helper_stage1_fit_nuisance(
  data,
  id_var,
  dp_var,
  outcome_var,
  treatment_var,
  mediator_var,
  avail_var,
  config_p,
  config_q,
  config_eta,
  config_mu,
  config_nu
)
```

Arguments

`data`	Data frame in long format.
`id_var`, `dp_var`, `outcome_var`, `treatment_var`, `mediator_var`, `avail_var`	Character column names (same as in `mcee_helper_2stage_estimation`).
`config_p`, `config_q`, `config_eta`, `config_mu`, `config_nu`	Configuration lists for each nuisance parameter (see `mcee_config_maker`).

Details

Nuisance Parameters Fitted:

- p: Propensity score $P(A_t = 1 \mid H_t)$ - fitted on available rows. (Technically, this is $P(A_t = I_t \mid H_t)$, but the user is allowed to input $P(A_t = 1 \mid H_t)$ and the function will automatically correct it by setting `p1 = 1` when $I_t = 0$.)
- q: Conditional propensity $P(A_t = 1 \mid H_t, M_t)$ - fitted on available rows. (Technically, this is $P(A_t = I_t \mid H_t, M_t)$, but the user is allowed to input $P(A_t = 1 \mid H_t, M_t)$ and the function will automatically correct it by setting `q1 = 1` when $I_t = 0$.)
- `eta1`, `eta0`: Outcome regression $E(Y \mid A_t = a, H_t)$ without mediator
- `mu1`, `mu0`: Outcome regression $E(Y \mid A_t = a, H_t, M_t)$ with mediator
- `nu1`, `nu0`: Cross-world regressions for counterfactual outcomes

Data Subsets Used for Fitting: - p, q: Only rows where `availability==1` - `eta1`, `mu1`: Rows where `A==1` OR `availability==0` - `eta0`, `mu0`: Rows where `A==0` - `nu1`: Fitted on `A==0` rows using `mu1` predictions as outcome - `nu0`: Fitted on `A==1` or unavailable rows using `mu0` predictions as outcome

Availability Handling: When `availability==0`, predictions are forced to `p1=p0=q1=q0=1` to prevent division by zero in Stage 2.

Value

List with two components:

`nuisance_fitted` List of numeric vectors (length `nrow(data)`) containing per-row predictions: `p1`, `p0`, `q1`, `q0`, `eta1`, `eta0`, `mu1`, `mu0`, `nu1`, `nu0`.

`nuisance_models` List of fitted model objects or "known" descriptors: `p`, `q`, `eta1`, `eta0`, `mu1`, `mu0`, `nu1`, `nu0`.

`mcee_helper_stage2_estimate_mcee`

Stage-2 MCEE parameter estimation given nuisance predictions

Description

Computes the Natural Direct Excursion Effect (NDEE; α) and Natural Indirect Excursion Effect (NIEE; β) parameters using Stage-1 nuisance predictions. This is Stage 2 of the two-stage MCEE procedure.

Usage

```
mcee_helper_stage2_estimate_mcee(
  data,
  id_var,
  dp_var,
  outcome_var,
```

```
  treatment_var,
  avail_var = NULL,
  p1,
  p0,
  q1,
  q0,
  eta1,
  eta0,
  mu1,
  mu0,
  nu1,
  nu0,
  omega_nrows,
  f_nrows
)
```

Arguments

`data`	Data frame in long format.
`id_var`, `dp_var`, `outcome_var`, `treatment_var`, `avail_var`	Character column names.
`p1`, `p0`, `q1`, `q0`, `eta1`, `eta0`, `mu1`, `mu0`, `nu1`, `nu0`	Numeric vectors of length `nrow(data)` containing Stage-1 nuisance predictions for each row.
`omega_nrows`	Numeric vector of length `nrow(data)` containing per-row weights $\omega(i,t) \geq 0$.
`f_nrows`	Numeric matrix with `nrow(data)` rows and `p` columns containing the basis functions $f(t)$ evaluated at each decision point.

Details

MCEE Estimating Equations: The function constructs influence functions ϕ_t^{11}, ϕ_t^{10}, ϕ_t^{00} for each row and solves the estimating equations:

- **NDEE (α)**: $\sum_{i,t} \omega(i,t)[\phi_t^{10} - \phi_t^{00}]f(t) = 0$
- **NIEE (β)**: $\sum_{i,t} \omega(i,t)[\phi_t^{11} - \phi_t^{10}]f(t) = 0$

Influence Functions: - ϕ_t^{11}: Direct effect pathway influence function - ϕ_t^{10}: Mediated effect pathway influence function - ϕ_t^{00}: Control/reference pathway influence function

Variance Estimation: Uses sandwich variance estimation with subject-level clustering. The variance accounts for the two-stage estimation uncertainty.

Value

List containing MCEE parameter estimates and inference:

`alpha_hat` Vector of length p: NDEE parameter estimates

`alpha_se` Vector of length p: NDEE standard errors

`beta_hat` Vector of length p: NIEE parameter estimates

`beta_se` Vector of length p: NIEE standard errors

`varcov` Matrix $2p \times 2p$: Joint variance-covariance for (α, β)

`alpha_varcov` Matrix $p \times p$: Variance-covariance for α

`beta_varcov` Matrix $p \times p$: Variance-covariance for β

`mcee_userfit_nuisance` *Mediated Causal Excursion Effects with user-supplied nuisance predictions*

Description

Skips Stage-1 model fitting and uses user-provided nuisance predictions.

Usage

```
mcee_userfit_nuisance(
  data,
  id,
  dp,
  outcome,
  treatment,
  mediator,
  availability = NULL,
  time_varying_effect_form,
  p1,
  q1,
  eta1,
  eta0,
  mu1,
  mu0,
  nu1,
  nu0,
  weight_per_row = NULL,
  verbose = TRUE
)
```

Arguments

`data`	A data.frame in long format (one row per id-by-decision point).
`id`	Character. Column name for subject identifier.
`dp`	Character. Column name for decision point index (must increase strictly within subject).
`outcome`	Character. Column name for distal outcome (constant within subject).
`treatment`	Character. Column name for treatment (coded 0/1).
`mediator`	Character. Column name for mediator.

availability Optional character. Column name for availability (0/1). If NULL, all rows are treated as available.

time_varying_effect_form
RHS-only formula for the basis $f(t)$ (e.g., ~ 1, ~ dp, ~ poly(dp,2)).

p1, q1, eta1, eta0, mu1, mu0, nu1, nu0
Numeric vectors (or column names) of per-row predictions aligned with data. See Details for definitions.

weight_per_row Optional numeric vector of row weights (nonnegative, length nrow(data)). If NULL, uniform within-id weights are used.

verbose Logical; print progress messages.

Details

Nuisance definitions:

- p1: $P(A_t = 1 \mid H_t)$ (known in MRTs). (Technically, this is $P(A_t = I_t \mid H_t)$, but the user is allowed to input $P(A_t = 1 \mid H_t)$ and the function will automatically correct it by setting p1 = 1 when $I_t = 0$.)
- q1: $P(A_t = 1 \mid H_t, M_t)$. (Technically, this is $P(A_t = I_t \mid H_t, M_t)$, but the user is allowed to input $P(A_t = 1 \mid H_t, M_t)$ and the function will automatically correct it by setting q1 = 1 when $I_t = 0$.)
- eta1, eta0: $E(Y \mid H_t, A_t = 1)$ and $E(Y \mid H_t, A_t = 0)$.
- mu1, mu0: $E(Y \mid H_t, A_t = 1, M_t)$ and $E(Y \mid H_t, A_t = 0, M_t)$.
- nu1, nu0: cross-world regressions; see vignette and paper for definitions.

If availability is provided, rows with $I = 0$ are coerced to p1=q1=1 (and hence p0=q0=1); a warning is emitted if overrides occur.

Value

An "mcee_fit" object; see mcee.

See Also

mcee, mcee_general

Examples

```
set.seed(1)
n <- 10
T <- 4
id <- rep(1:n, each = T)
dp <- rep(1:T, times = n)
A <- rbinom(n * T, 1, 0.5)
M <- rbinom(n * T, 1, plogis(-0.2 + 0.3 * A + 0.1 * dp))
Y <- ave(0.5 * A + 0.6 * M + 0.1 * dp + rnorm(n * T), id)
dat <- data.frame(id, dp, A, M, Y)

fit_usr <- mcee_userfit_nuisance(dat, "id","dp","Y","A","M",
```

```
time_varying_effect_form = ~ dp,
p1 = rep(0.5, nrow(dat)),
q1 = runif(nrow(dat),.3,.7),
eta1 = rnorm(nrow(dat)), eta0 = rnorm(nrow(dat)),
mu1 = rnorm(nrow(dat)),  mu0 = rnorm(nrow(dat)),
nu1 = rnorm(nrow(dat)),  nu0 = rnorm(nrow(dat)))
```

`print.summary.mcee_fit` *Print method for summary of MCEE fits*

Description

Prints formatted coefficient tables and inference results for mediated causal excursion effects, including alpha (Natural Direct Excursion Effect) and beta (Natural Indirect Excursion Effect) parameters.

Usage

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

Arguments

`x`	An object of class `"summary.mcee_fit"` from `summary.mcee_fit`.
`...`	Currently unused.

Value

Invisibly returns the input object `x`. Called for side effects.

See Also

`summary.mcee_fit`

`summary.dcee_fit` *Summary for DCEE fits*

Description

Produce inference tables for distal causal excursion effects from a [dcee()] model. By default uses small-sample t-tests with `df = object$df` (subjects minus number of betas). If `df` is missing or nonpositive, falls back to large-sample normal (z) inference.

Usage

```
## S3 method for class 'dcee_fit'
summary(
  object,
  lincomb = NULL,
  conf_level = 0.95,
  show_control_fit = FALSE,
  ...
)
```

Arguments

object	An object of class "dcee_fit" returned by [dcee()].
lincomb	Optional numeric vector or matrix specifying linear combinations $L\beta$. If a vector of length p (number of betas), a single linear combination is evaluated. If a matrix, it must have p columns; each row defines one combination. Row names (if present) are used as labels.
conf_level	Confidence level for intervals (default 0.95).
show_control_fit	Logical; if TRUE, include compact information about the Stage-1 nuisance regressions (if available). When cross_fit = TRUE in [dcee()], regfit_a0/regfit_a1 refer to the *last fold* fit and are provided for inspection only.
...	Currently ignored.

Value

A list of class "summary.dcee_fit" with components:

- call — the original call
- df — degrees of freedom used for t-tests (may be NA)
- conf_level — the confidence level
- excursion_effect — data frame with coefficient table for β
- lincomb — optional data frame with linear-combination results
- control_fit — optional list describing Stage-1 fits (only if show_control_fit)

---

summary.emee_fit *Summarize Causal Excursion Effect Fits for MRT with Binary Outcomes*

Description

summary method for class "emee_fit".

Usage

```
## S3 method for class 'emee_fit'
summary(
  object,
  lincomb = NULL,
  conf_level = 0.95,
  show_control_fit = FALSE,
  ...
)
```

Arguments

object	An object of class "emee_fit".
lincomb	A vector of length p (p is the number of moderators including intercept) or a matrix with p columns. When not set to 'NULL', the summary will include the specified linear combinations of the causal excursion effect coefficients and the corresponding confidence interval, standard error, and p-value.
conf_level	A numeric value indicating the confidence level for confidence intervals. Default to 0.95.
show_control_fit	A logical value of whether the fitted coefficients for the control variables will be printed in the summary. Default to FALSE. (Interpreting the fitted coefficients for control variables is not recommended.)
...	Further arguments passed to or from other methods.

Value

the original function call and the estimated causal excursion effect coefficients, confidence interval with conf_level, standard error, t-statistic value, degrees of freedom, and p-value.

Examples

```
fit <- emee(
    data = data_binary,
    id = "userid",
    outcome = "Y",
    treatment = "A",
    rand_prob = "rand_prob",
    moderator_formula = ~time_var1,
    control_formula = ~ time_var1 + time_var2,
    availability = "avail",
    numerator_prob = 0.5,
    start = NULL
)
summary(fit)
```

`summary.mcee_fit` *Summarize an mcee fit*

Description

Prints coefficient tables for the Natural Direct Excursion Effect (alpha) and Natural Indirect Excursion Effect (beta), with small-sample t inference. Optionally reports linear combinations and Stage-1 nuisance summaries.

Usage

```
## S3 method for class 'mcee_fit'
summary(
  object,
  lincomb_alpha = NULL,
  lincomb_beta = NULL,
  lincomb_joint = NULL,
  conf_level = 0.95,
  show_nuisance = FALSE,
  ...
)
```

Arguments

`object`	An object of class `"mcee_fit"`.
`lincomb_alpha, lincomb_beta`	Optional numeric vector or matrix specifying linear combinations of `alpha` or `beta` coefficients.
`lincomb_joint`	Optional numeric vector or matrix specifying linear combinations of the stacked parameter `c(alpha, beta)`.
`conf_level`	Confidence level for Wald intervals (default 0.95).
`show_nuisance`	Logical; if `TRUE`, prints a compact summary of Stage-1 nuisance fits.
`...`	Unused.

Value

A list of class `"summary.mcee_fit"` with printed side effects.

Examples

```
# s <- summary(fit, lincomb_alpha = c(1, 9), lincomb_beta = c(1, 9))
```

summary.wcls_fit *Summarize Causal Excursion Effect Fits for MRT with Continuous Outcomes*

Description

summary method for class "wcls_fit".

Usage

```
## S3 method for class 'wcls_fit'
summary(
  object,
  lincomb = NULL,
  conf_level = 0.95,
  show_control_fit = FALSE,
  ...
)
```

Arguments

object	An object of class "wcls_fit".
lincomb	A vector of length p (p is the number of moderators including intercept) or a matrix with p columns. When not set to 'NULL', the summary will include the specified linear combinations of the causal excursion effect coefficients and the corresponding confidence interval, standard error, and p-value.
conf_level	A numeric value indicating the confidence level for confidence intervals. Default to 0.95.
show_control_fit	A logical value of whether the fitted coefficients for the control variables will be printed in the summary. Default to FALSE. (Interpreting the fitted coefficients for control variables is not recommended.)
...	Further arguments passed to or from other methods.

Value

the original function call and the estimated causal excursion effect coefficients, 95 value or Wald-statistic value (depending on whether sample size is < 50), degrees of freedom, and p-value.

Examples

```
fit <- wcls(
    data = data_mimicHeartSteps,
    id = "userid",
    outcome = "logstep_30min",
    treatment = "intervention",
    rand_prob = 0.6,
```

```
    moderator_formula = ~1,
    control_formula = ~logstep_pre30min,
    availability = "avail",
    numerator_prob = 0.6
)
summary(fit)
```

`wcls` *Estimates the causal excursion effect for continuous outcome MRT*

Description

Returns the estimated causal excursion effect (on additive scale) and the estimated standard error. Small sample correction using the "Hat" matrix in the variance estimate is implemented.

Usage

```
wcls(
  data,
  id,
  outcome,
  treatment,
  rand_prob,
  moderator_formula,
  control_formula,
  availability = NULL,
  numerator_prob = NULL,
  verbose = TRUE
)
```

Arguments

`data`	A data set in long format.
`id`	The subject id variable.
`outcome`	The outcome variable.
`treatment`	The binary treatment assignment variable.
`rand_prob`	The randomization probability variable.
`moderator_formula`	A formula for the moderator variables. This should start with ~ followed by the moderator variables. When set to `~ 1`, a fully marginal excursion effect (no moderators) is estimated.
`control_formula`	A formula for the control variables. This should start with ~ followed by the control variables. When set to `~ 1`, only an intercept is included as the control variable.

availability	The availability variable. Use the default value (NULL) if your MRT doesn't have availability considerations.
numerator_prob	Either a number between 0 and 1, or a variable name for a column in data. If you are not sure what this is, use the default value (NULL).
verbose	If default ('TRUE'), additional messages will be printed during data preprocessing.

Value

An object of type "wcls_fit"

Examples

```
wcls(
    data = data_mimicHeartSteps,
    id = "userid",
    outcome = "logstep_30min",
    treatment = "intervention",
    rand_prob = 0.6,
    moderator_formula = ~1,
    control_formula = ~logstep_pre30min,
    availability = "avail",
    numerator_prob = 0.6
)
```

Index