

Package ‘mixedsubjects’

July 2, 2026

Title Causal Inference in Experiments with Mixed-Subjects Designs

Version 1.0.0

Description Implements seven estimators for average treatment effect (ATE) estimation in mixed-subjects designs (MSDs), where human subjects data is augmented with predictions from large language models (LLMs). Includes Difference-in-Means, GREG, PPI++, Doubly-Tuned, Difference-in-Predictions (DiP), DiP++, and D-T DiP estimators. Provides point estimates, variance estimation via delta-method or bootstrap, and optimal design selection for budget allocation between human observations and LLM predictions.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://klinikkanopka.com/mixedsubjects/>

BugReports <https://github.com/klinikkanopka/mixedsubjects/issues>

NeedsCompilation no

Author Austin van Loon [aut],
Klint Kanopka [aut, cre],
Yuan Huang [ctb]

Maintainer Klint Kanopka <klinik.kanopka@nyu.edu>

Repository CRAN

Date/Publication 2026-07-02 18:30:02 UTC

Contents

| | |
|------------------------------------|---|
| bootstrap_variance | 2 |
| compare_variance_methods | 4 |

| | |
|--------------------------|----|
| estimate_all | 5 |
| msd_data | 5 |
| msd_dim | 8 |
| msd_dip | 10 |
| msd_dip_pp | 12 |
| msd_dt | 13 |
| msd_dt_dip | 15 |
| msd_greg | 17 |
| msd_ppi | 19 |
| optimal_design | 21 |
| print.msd_data | 23 |
| print.msd_design | 24 |
| print.msd_result | 24 |
| print.msd_summary | 25 |
| print.summary.msd_result | 25 |
| summary.msd_design | 26 |
| summary.msd_result | 26 |

| | |
|--------------|-----------|
| Index | 28 |
|--------------|-----------|

| | |
|--------------------|--|
| bootstrap_variance | <i>Variance Estimation for Mixed-Subjects Design</i> |
|--------------------|--|

Description

Variance estimation methods including fold-respecting bootstrap. Bootstrap Variance Estimation

Computes bootstrap variance estimates for MSD estimators using a fold-respecting resampling procedure.

Usage

```
bootstrap_variance(
  data,
  estimator = c("dim", "greg", "ppi", "dt", "dip", "dip_pp", "dt_dip"),
  n_bootstrap = 1000,
  n_folds = 2,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

| | |
|-------------|--|
| data | An <code>msd_data</code> object created by msd_data |
| estimator | Character string specifying the estimator. One of: "dim", "greg", "ppi", "dt", "dip", "dip_pp", "dt_dip" |
| n_bootstrap | Number of bootstrap replications (default 1000) |
| n_folds | Number of folds for cross-fitting (default 2, used for tuned estimators) |

| | |
|------------|--|
| conf_level | Confidence level for confidence intervals (default 0.95) |
| seed | Random seed for reproducibility (optional) |

Details

The fold-respecting bootstrap resamples within each stratum:

1. Resample observed treatment units with replacement
2. Resample observed control units with replacement
3. Resample unlabeled units with replacement
4. Recompute the estimator on the bootstrap sample

For cross-fit estimators, the fold assignments are regenerated for each bootstrap replicate to properly account for the cross-fitting variance.

Value

A list containing:

| | |
|---------------------|--|
| estimate | Point estimate from the original data |
| variance | Bootstrap variance estimate |
| se | Bootstrap standard error |
| ci_lower, ci_upper | Bootstrap percentile confidence interval |
| bootstrap_estimates | Vector of bootstrap estimates |

Examples

```
# Create sample data
obs_df <- data.frame(
  Y = rnorm(100),
  S0 = rnorm(100),
  S1 = rnorm(100),
  D = rep(c(1, 0), each = 50)
)
unobs_df <- data.frame(
  S0 = rnorm(200),
  S1 = rnorm(200),
  D = rep(c(1, 0), each = 100)
)
msd <- msd_data(observed = obs_df, unobserved = unobs_df)

# Bootstrap variance for D-T DiP
boot_result <- bootstrap_variance(msd, "dt_dip", n_bootstrap = 100, seed = 1)
print(boot_result)
```

`compare_variance_methods`*Compare variance estimates across methods*

Description

Computes and compares variance estimates using both delta-method and bootstrap for a given estimator.

Usage

```
compare_variance_methods(  
  data,  
  estimator,  
  n_bootstrap = 1000,  
  n_folds = 2,  
  seed = NULL  
)
```

Arguments

| | |
|--------------------------|---|
| <code>data</code> | An <code>msd_data</code> object |
| <code>estimator</code> | Character string specifying the estimator |
| <code>n_bootstrap</code> | Number of bootstrap replications |
| <code>n_folds</code> | Number of folds for cross-fitting |
| <code>seed</code> | Random seed |

Value

A data frame comparing variance estimates

Examples

```
obs_df <- data.frame(  
  Y = rnorm(100), S0 = rnorm(100), S1 = rnorm(100),  
  D = rep(c(1, 0), each = 50)  
)  
unobs_df <- data.frame(  
  S0 = rnorm(200), S1 = rnorm(200), D = rep(c(1, 0), each = 100)  
)  
msd <- msd_data(observed = obs_df, unobserved = unobs_df)  
comparison <- compare_variance_methods(msd, "dt_dip", n_bootstrap = 100, seed = 1)  
print(comparison)
```

| | |
|--------------|--|
| estimate_all | <i>Estimate all available estimators</i> |
|--------------|--|

Description

Runs all applicable estimators on the data and returns a summary table.

Usage

```
estimate_all(data, n_folds = 2, conf_level = 0.95)
```

Arguments

| | |
|------------|---|
| data | An msd_data object |
| n_folds | Number of folds for cross-fitting (default 2) |
| conf_level | Confidence level (default 0.95) |

Value

A data frame with estimates from all applicable estimators

Examples

```
obs_df <- data.frame(
  Y = rnorm(100), S0 = rnorm(100), S1 = rnorm(100),
  D = rep(c(1, 0), each = 50)
)
unobs_df <- data.frame(
  S0 = rnorm(200), S1 = rnorm(200), D = rep(c(1, 0), each = 100)
)
msd <- msd_data(observed = obs_df, unobserved = unobs_df)
all_estimates <- estimate_all(msd)
print(all_estimates)
```

| | |
|----------|---|
| msd_data | <i>Data Preparation for Mixed-Subjects Design</i> |
|----------|---|

Description

Create and validate data objects for MSD estimation. Create an MSD data object

Creates a validated data object for mixed-subjects design estimation. Accepts either a single combined dataframe or two separate dataframes for observed and unobserved units. Column names default to "Y", "D", "S0", "S1" and can be overridden explicitly.

Usage

```
msd_data(
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  outcome = "Y",
  treatment = "D",
  pred_control = "S0",
  pred_treated = "S1",
  obs_outcome = NULL,
  obs_treatment = NULL,
  obs_pred_control = NULL,
  obs_pred_treated = NULL,
  unobs_treatment = NULL,
  unobs_pred_control = NULL,
  unobs_pred_treated = NULL
)
```

Arguments

| | |
|---------------------------------|--|
| <code>data</code> | A combined dataframe containing both observed and unobserved units. Observed units have non-missing Y values; unobserved units have Y = NA. If provided, observed and unobserved should be NULL. |
| <code>observed</code> | A dataframe of observed (labeled) units with columns for outcome Y, predictions S0/S1, and treatment D. If provided, data should be NULL. |
| <code>unobserved</code> | A dataframe of unobserved (unlabeled) units with columns for predictions S0/S1 and treatment D (no Y column needed). |
| <code>outcome</code> | Name of the outcome column. Default: "Y". |
| <code>treatment</code> | Name of the treatment column. Default: "D". |
| <code>pred_control</code> | Name of the control prediction column. Default: "S0". |
| <code>pred_treated</code> | Name of the treatment prediction column. Default: "S1". |
| <code>obs_outcome</code> | Outcome column name for observed data only (overrides outcome). |
| <code>obs_treatment</code> | Treatment column name for observed data only (overrides treatment). |
| <code>obs_pred_control</code> | Control prediction column for observed data only. |
| <code>obs_pred_treated</code> | Treatment prediction column for observed data only. |
| <code>unobs_treatment</code> | Treatment column name for unobserved data only. |
| <code>unobs_pred_control</code> | Control prediction column for unobserved data only. |
| <code>unobs_pred_treated</code> | Treatment prediction column for unobserved data only. |

Details

The function supports flexible column name specification:

Default column names: By default, the function expects columns named "Y" (outcome), "D" (treatment), "S0" (control prediction), and "S1" (treatment prediction). Override these using the `outcome`, `treatment`, `pred_control`, and `pred_treated` arguments.

Global specification: Use `outcome`, `treatment`, `pred_control`, `pred_treated` to set column names that apply to both observed and unobserved dataframes.

Per-dataframe specification: Use `obs_*` and `unobs_*` arguments when column names differ between observed and unobserved dataframes. These override global settings.

Two input modes:

Mode 1: Single combined dataframe Provide a single dataframe via the `data` argument. The function will automatically split it into observed and unobserved based on whether Y is NA.

Mode 2: Separate dataframes Provide two separate dataframes via `observed` and `unobserved`.

Value

An S3 object of class "msd_data" containing:

| | |
|-----------------------------------|---|
| <code>observed</code> | Dataframe of observed units with standardized columns Y, S0, S1, D |
| <code>unobserved</code> | Dataframe of unobserved units with standardized columns S0, S1, D (or NULL) |
| <code>has_S0</code> | Logical indicating if S0 predictions are available |
| <code>has_S1</code> | Logical indicating if S1 predictions are available |
| <code>has_both_predictions</code> | Logical indicating if both S0 and S1 are available |
| <code>n1</code> | Number of treated observed units |
| <code>n0</code> | Number of control observed units |
| <code>m</code> | Number of unobserved units |
| <code>col_mapping</code> | List of original column names used |

Examples

```
# Default column names (Y, D, S0, S1)
obs_df <- data.frame(
  Y = c(1.2, 0.8, 1.5, 0.9),
  S0 = c(1.0, 0.7, 1.3, 0.8),
  S1 = c(1.1, 0.9, 1.4, 1.0),
  D = c(1, 0, 1, 0)
)
msd <- msd_data(observed = obs_df)

# Custom column names (same in both dataframes)
obs_df2 <- data.frame(
  response = c(1.2, 0.8, 1.5, 0.9),
  pred_ctrl = c(1.0, 0.7, 1.3, 0.8),
  pred_trt = c(1.1, 0.9, 1.4, 1.0),
  treated = c(1, 0, 1, 0)
)
```

```

)
unobs_df2 <- data.frame(
  pred_ctrl1 = c(1.1, 0.9),
  pred_trt = c(1.2, 1.0),
  treated = c(1, 0)
)
msd2 <- msd_data(
  observed = obs_df2,
  unobserved = unobs_df2,
  outcome = "response",
  treatment = "treated",
  pred_control = "pred_ctrl1",
  pred_treated = "pred_trt"
)

# Different column names in observed vs unobserved
obs_df3 <- data.frame(
  outcome = c(1.2, 0.8),
  claude_pred_0 = c(1.0, 0.7),
  claude_pred_1 = c(1.1, 0.9),
  treatment = c(1, 0)
)
unobs_df3 <- data.frame(
  s0_claude = c(1.1, 0.9),
  s1_claude = c(1.2, 1.0),
  D = c(1, 0)
)
msd3 <- msd_data(
  observed = obs_df3,
  unobserved = unobs_df3,
  obs_outcome = "outcome",
  obs_treatment = "treatment",
  obs_pred_control = "claude_pred_0",
  obs_pred_treated = "claude_pred_1",
  unobs_treatment = "D",
  unobs_pred_control = "s0_claude",
  unobs_pred_treated = "s1_claude"
)

```

msd_dim

Difference-in-Means Estimator

Description

Classical difference-in-means estimator for ATE. Difference-in-Means Estimator

Computes the classical difference-in-means estimator for the average treatment effect (ATE). This estimator uses only observed (labeled) data and does not incorporate any predictions.

Usage

```
msd_dim(
  formula_or_data,
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  conf_level = 0.95
)
```

Arguments

| | |
|-----------------|--|
| formula_or_data | Either an msd_data object created by <code>msd_data</code> , or a formula of the form <code>outcome ~ treatment</code> (predictions not needed for DiM). |
| data | If formula_or_data is a formula, this should be either: an msd_data object, a combined dataframe, or NULL (if using observed/unobserved). |
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| conf_level | Confidence level for the confidence interval (default 0.95) |

Details

The difference-in-means estimator is:

$$\hat{\tau}^{DiM} = \bar{Y}_1 - \bar{Y}_0$$

where \bar{Y}_d is the sample mean of outcomes in arm d .

The variance is estimated as:

$$\widehat{Var}(\hat{\tau}^{DiM}) = \frac{s_{Y(1)}^2}{n_1} + \frac{s_{Y(0)}^2}{n_0}$$

where $s_{Y(d)}^2$ is the sample variance of outcomes in arm d .

Value

An `msd_result` object containing:

| | |
|--------------------|---|
| estimate | Point estimate of the ATE: <code>mean(YID=1) - mean(YID=0)</code> |
| variance | Estimated variance: <code>var(YID=1)/n1 + var(YID=0)/n0</code> |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |

Examples

```
# Using msd_data object (standard interface)
obs_df <- data.frame(
  Y = c(1.2, 1.4, 0.8, 0.6, 1.1, 0.9, 1.3, 0.7),
  S0 = c(1.0, 1.2, 0.7, 0.5, 1.0, 0.8, 1.1, 0.6),
  S1 = c(1.1, 1.3, 0.9, 0.7, 1.1, 0.9, 1.2, 0.8),
  D = c(1, 1, 0, 0, 1, 0, 1, 0)
)
msd <- msd_data(observed = obs_df)

result <- msd_dim(msd)
print(result)

# Using formula interface with custom column names
df <- data.frame(
  response = c(1.2, 1.4, 0.8, 0.6),
  treated = c(1, 1, 0, 0)
)
result2 <- msd_dim(response ~ treated, observed = df)
```

msd_dip

*DiP (Difference-in-Predictions) Estimator***Description**

DiP estimator for ATE using paired predictions. DiP (Difference-in-Predictions) Estimator

Computes the Difference-in-Predictions (DiP) estimator for the average treatment effect (ATE). This estimator uses both treatment and control predictions for each unlabeled unit, computing the contrast $S^{(1)} - S^{(0)}$ at the unit level.

Usage

```
msd_dip(
  formula_or_data,
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  conf_level = 0.95
)
```

Arguments

formula_or_data Either an `msd_data` object created by `msd_data`, or a formula of the form `outcome ~ treatment | pred_treated + pred_control`.

data If `formula_or_data` is a formula, this should be either: an `msd_data` object, a combined dataframe, or `NULL` (if using `observed/unobserved`).

| | |
|------------|---|
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| conf_level | Confidence level for the confidence interval (default 0.95) |

Details

The DiP estimator is:

$$\hat{\tau}^{DiP} = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} (S_i^{(1)} - S_i^{(0)}) + \frac{1}{n_1} \sum_{i \in \mathcal{O}_1} (Y_i - S_i^{(1)}) - \frac{1}{n_0} \sum_{i \in \mathcal{O}_0} (Y_i - S_i^{(0)})$$

Value

An msd_result object containing:

| | |
|--------------------|-------------------------------------|
| estimate | Point estimate of the ATE |
| variance | Estimated variance |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |
| lambda | Tuning parameter (always 1 for DiP) |

Note

DiP requires BOTH S0 and S1 predictions for ALL units. The key advantage of DiP over GREG is that when $S^{(1)}$ and $S^{(0)}$ are positively correlated, the variance of their difference is smaller.

Examples

```
# Using msd_data object
obs_df <- data.frame(
  Y = c(1.2, 1.4, 0.8, 0.6),
  S0 = c(1.0, 1.2, 0.7, 0.5),
  S1 = c(1.1, 1.3, 0.9, 0.7),
  D = c(1, 1, 0, 0)
)
unobs_df <- data.frame(
  S0 = c(1.1, 0.9, 1.0, 0.8),
  S1 = c(1.2, 1.0, 1.1, 0.9),
  D = c(1, 1, 0, 0)
)
msd <- msd_data(observed = obs_df, unobserved = unobs_df)
result <- msd_dip(msd)

# Using formula interface
result2 <- msd_dip(Y ~ D | S1 + S0, observed = obs_df, unobserved = unobs_df)
```

msd_dip_pp

*DiP++ Estimator***Description**

Power-tuned Difference-in-Predictions estimator for ATE. DiP++ Estimator

Computes the DiP++ (power-tuned difference-in-predictions) estimator for the average treatment effect (ATE). This estimator uses paired predictions $S^{(1)}$ and $S^{(0)}$ for each unlabeled unit, with a single tuning parameter λ estimated via cross-fitting.

Usage

```
msd_dip_pp(
  formula_or_data,
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  n_folds = 2,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

| | |
|-----------------|--|
| formula_or_data | Either an <code>msd_data</code> object created by <code>msd_data</code> , or a formula of the form <code>outcome ~ treatment pred_treated + pred_control</code> . |
| data | If <code>formula_or_data</code> is a formula, this should be either: an <code>msd_data</code> object, a combined dataframe, or <code>NULL</code> (if using <code>observed/unobserved</code>). |
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| n_folds | Number of folds for cross-fitting (default 2) |
| conf_level | Confidence level for the confidence interval (default 0.95) |
| seed | Random seed for fold splitting (optional) |

Details

The DiP++ estimator is:

$$\hat{\tau}^{DiP++}(\lambda) = \frac{\lambda}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} (S_i^{(1)} - S_i^{(0)}) + \frac{1}{n_1} \sum_{i \in \mathcal{O}_1} (Y_i - \lambda S_i^{(1)}) - \frac{1}{n_0} \sum_{i \in \mathcal{O}_0} (Y_i - \lambda S_i^{(0)})$$

Value

An msd_result object containing:

| | |
|--------------------|---|
| estimate | Point estimate of the ATE |
| variance | Estimated variance (delta-method) |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |
| lambda | Estimated tuning parameter (single value) |

Note

DiP++ requires BOTH S0 and S1 predictions for ALL units. For arm-specific tuning, use [msd_dt_dip](#).

Examples

```
# Using msd_data object
set.seed(123)
n <- 100
obs_df <- data.frame(
  Y = rnorm(n),
  D = rep(c(1, 0), each = n/2)
)
obs_df$Y <- obs_df$Y + 0.3 * obs_df$D
obs_df$S1 <- 0.5 * obs_df$Y + rnorm(n, 0, 0.5)
obs_df$S0 <- 0.5 * obs_df$Y + rnorm(n, 0, 0.5) - 0.1

unobs_df <- data.frame(
  S0 = rnorm(200, 0, 0.5),
  S1 = rnorm(200, 0.2, 0.5),
  D = rep(c(1, 0), each = 100)
)

msd <- msd_data(observed = obs_df, unobserved = unobs_df)
result <- msd_dip_pp(msd)

# Using formula interface
result2 <- msd_dip_pp(Y ~ D | S1 + S0, observed = obs_df, unobserved = unobs_df)
```

msd_dt

D-T (Doubly-Tuned) Estimator

Description

D-T estimator with arm-specific tuning parameters for ATE. D-T (Doubly-Tuned) Estimator
 Computes the Doubly-Tuned (D-T) estimator for the average treatment effect (ATE). This estimator uses arm-specific tuning parameters (lambda_1 and lambda_0) estimated via cross-fitting.

Usage

```
msd_dt(
  formula_or_data,
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  n_folds = 2,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

| | |
|-----------------|---|
| formula_or_data | Either an msd_data object created by <code>msd_data</code> , or a formula of the form <code>outcome ~ treatment prediction</code> . |
| data | If formula_or_data is a formula, this should be either: an msd_data object, a combined dataframe, or NULL (if using observed/unobserved). |
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| n_folds | Number of folds for cross-fitting (default 2) |
| conf_level | Confidence level for the confidence interval (default 0.95) |
| seed | Random seed for fold splitting (optional) |

Details

The D-T estimator uses arm-specific tuning parameters:

$$\hat{\mu}_d^{D-T}(\lambda_d) = \bar{Y}_{O_d} + \lambda_d(\bar{S}_{U_d}^{(d)} - \bar{S}_{O_d}^{(d)})$$

Each λ_{d} is chosen to minimize the variance in arm d :

$$\lambda_d^* = \frac{Cov(Y(d), S^{(d)})/n_d}{Var(S^{(d)})(1/m_d + 1/n_d)}$$

The tuning parameters are estimated via cross-fitting to avoid bias.

Value

An `msd_result` object containing:

| | |
|--------------------|--|
| estimate | Point estimate of the ATE |
| variance | Estimated variance (delta-method) |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |
| lambda | Vector of arm-specific tuning parameters (<code>lambda_1</code> , <code>lambda_0</code>) |

Note

D-T differs from PPI++ by using separate tuning parameters for each arm, which can improve efficiency when the prediction quality differs between treatment and control.

Examples

```
# Create sample data
set.seed(123)
n <- 100
obs_df <- data.frame(
  Y = rnorm(n),
  S0 = rnorm(n, 0, 0.5),
  S1 = rnorm(n, 0.2, 0.5),
  D = rep(c(1, 0), each = n/2)
)
obs_df$Y <- obs_df$Y + 0.3 * obs_df$D
obs_df$S1[obs_df$D == 1] <- obs_df$S1[obs_df$D == 1] + 0.5 * obs_df$Y[obs_df$D == 1]
obs_df$S0[obs_df$D == 0] <- obs_df$S0[obs_df$D == 0] + 0.5 * obs_df$Y[obs_df$D == 0]

unobs_df <- data.frame(
  S0 = rnorm(200, 0, 0.5),
  S1 = rnorm(200, 0.2, 0.5),
  D = rep(c(1, 0), each = 100)
)

msd <- msd_data(observed = obs_df, unobserved = unobs_df)
result <- msd_dt(msd)

# Using formula interface
result2 <- msd_dt(Y ~ D | S1 + S0, observed = obs_df, unobserved = unobs_df)
```

msd_dt_dip

*D-T DiP (Doubly-Tuned Difference-in-Predictions) Estimator***Description**

D-T DiP estimator with arm-specific tuning for ATE. D-T DiP (Doubly-Tuned Difference-in-Predictions) Estimator

Computes the D-T DiP estimator for the average treatment effect (ATE). This estimator uses paired predictions $S^{(1)}$ and $S^{(0)}$ for each unlabeled unit, with arm-specific tuning parameters (λ_1 and λ_0) estimated via cross-fitting.

Usage

```
msd_dt_dip(
  formula_or_data,
  data = NULL,
```

```

observed = NULL,
unobserved = NULL,
n_folds = 2,
conf_level = 0.95,
seed = NULL
)

```

Arguments

| | |
|-----------------|--|
| formula_or_data | Either an msd_data object created by <code>msd_data</code> , or a formula of the form <code>outcome ~ treatment pred_treated + pred_control</code> . |
| data | If <code>formula_or_data</code> is a formula, this should be either: an msd_data object, a combined dataframe, or NULL (if using <code>observed/unobserved</code>). |
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| n_folds | Number of folds for cross-fitting (default 2) |
| conf_level | Confidence level for the confidence interval (default 0.95) |
| seed | Random seed for fold splitting (optional) |

Details

The D-T DiP estimator is:

$$\hat{\tau}^{D-TDiP} = \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} (\lambda_1 S_i^{(1)} - \lambda_0 S_i^{(0)}) + \frac{1}{n_1} \sum_{i \in \mathcal{O}_1} (Y_i - \lambda_1 S_i^{(1)}) - \frac{1}{n_0} \sum_{i \in \mathcal{O}_0} (Y_i - \lambda_0 S_i^{(0)})$$

The tuning parameters (λ_1, λ_0) are chosen jointly to minimize the total variance. Because the two arms share the unobserved pool, the variance is jointly quadratic in (λ_1, λ_0) with a cross-term from $Cov(S^{(1)}, S^{(0)})$, so the optimum solves a coupled 2x2 linear system rather than two independent regressions.

The tuning parameters are estimated via cross-fitting:

1. Split labeled data into K folds
2. For each fold k, estimate $(\lambda_{1k}, \lambda_{0k})$ on opposite folds
3. Compute the fold-k estimate using estimated lambdas
4. Average across folds with equal weights

Value

An `msd_result` object containing:

| | |
|--------------------|---|
| estimate | Point estimate of the ATE |
| variance | Estimated variance (delta-method) |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |
| lambda | Vector of arm-specific tuning parameters $(\lambda_{1k}, \lambda_{0k})$ |

Note

D-T DiP requires BOTH S_0 and S_1 predictions for ALL units. This corresponds to 2 predictions per unlabeled unit.

D-T DiP combines the benefits of:

- DiP: exploiting positive correlation between $S^{(1)}$ and $S^{(0)}$
- D-T: arm-specific tuning for heterogeneous prediction quality

Examples

```
# Create sample data with both predictions
set.seed(123)
n <- 100
obs_df <- data.frame(
  Y = rnorm(n),
  D = rep(c(1, 0), each = n/2)
)
obs_df$Y <- obs_df$Y + 0.3 * obs_df$D
obs_df$S1 <- 0.6 * obs_df$Y + rnorm(n, 0, 0.4)
obs_df$S0 <- 0.4 * obs_df$Y + rnorm(n, 0, 0.5)

unobs_df <- data.frame(
  S0 = rnorm(300, 0, 0.5),
  S1 = rnorm(300, 0.2, 0.4),
  D = rep(c(1, 0), 150)
)
# Add correlation between S0 and S1
unobs_df$S1 <- unobs_df$S1 + 0.5 * unobs_df$S0

msd <- msd_data(observed = obs_df, unobserved = unobs_df)
result <- msd_dt_dip(msd)

# Using formula interface
result2 <- msd_dt_dip(Y ~ D | S1 + S0, observed = obs_df, unobserved = unobs_df)
```

msd_greg

GREG Estimator

Description

Generalized Regression (GREG) calibration estimator for ATE. GREG Estimator

Computes the Generalized Regression (GREG) calibration estimator for the average treatment effect (ATE). This is the untuned calibration estimator with tuning parameter $\lambda = 1$. GREG, the original Prediction-Powered Inference (PPI) estimator, and Design-based Supervised Learning (DSL) are all the same untuned ($\lambda = 1$) estimator. It is distinct from the power-tuned PPI++ estimator in [msd_ppi](#), which estimates λ via cross-fitting to minimize variance.

Usage

```
msd_greg(
  formula_or_data,
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  conf_level = 0.95
)
```

Arguments

| | |
|-----------------|--|
| formula_or_data | Either an msd_data object created by <code>msd_data</code> , or a formula of the form <code>outcome ~ treatment prediction</code> . For GREG, the formula specifies which prediction column(s) to use. |
| data | If <code>formula_or_data</code> is a formula, this should be either: an msd_data object, a combined dataframe, or NULL (if using <code>observed/unobserved</code>). |
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| conf_level | Confidence level for the confidence interval (default 0.95) |

Details

The GREG estimator for arm d is:

$$\hat{\mu}_d^{GREG} = \bar{S}_{U_d}^{(d)} + (\bar{Y}_{O_d} - \bar{S}_{O_d}^{(d)})$$

The ATE estimate is:

$$\hat{\tau}^{GREG} = \hat{\mu}_1^{GREG} - \hat{\mu}_0^{GREG}$$

The variance is:

$$\widehat{Var}(\hat{\tau}^{GREG}) = \sum_{d \in \{0,1\}} \left[\frac{s_{S^{(d)}}^2}{m_d} + \frac{Var(Y(d) - S^{(d)})}{n_d} \right]$$

Value

An `msd_result` object containing:

| | |
|--------------------|--------------------------------------|
| estimate | Point estimate of the ATE |
| variance | Estimated variance |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |
| lambda | Tuning parameter (always 1 for GREG) |

Note

GREG requires predictions for each unit's assigned arm:

- Treatment arm units need S1
- Control arm units need S0

Examples

```
# Using msd_data object
obs_df <- data.frame(
  Y = c(1.2, 1.4, 0.8, 0.6),
  S0 = c(1.0, 1.2, 0.7, 0.5),
  S1 = c(1.1, 1.3, 0.9, 0.7),
  D = c(1, 1, 0, 0)
)
unobs_df <- data.frame(
  S0 = c(1.1, 0.9, 1.0, 0.8),
  S1 = c(1.2, 1.0, 1.1, 0.9),
  D = c(1, 1, 0, 0)
)
msd <- msd_data(observed = obs_df, unobserved = unobs_df)
result <- msd_greg(msd)

# Using formula interface
result2 <- msd_greg(Y ~ D | S1 + S0, observed = obs_df, unobserved = unobs_df)
```

msd_ppi

PPI++ Estimator

Description

Power-tuned Prediction-Powered Inference estimator for ATE. PPI++ Estimator

Computes the PPI++ (power-tuned prediction-powered inference) estimator for the average treatment effect (ATE). This estimator uses a single tuning parameter lambda that is estimated via cross-fitting to minimize variance.

Usage

```
msd_ppi(
  formula_or_data,
  data = NULL,
  observed = NULL,
  unobserved = NULL,
  n_folds = 2,
  conf_level = 0.95,
  seed = NULL
)
```

Arguments

| | |
|-----------------|---|
| formula_or_data | Either an msd_data object created by <code>msd_data</code> , or a formula of the form <code>outcome ~ treatment prediction</code> . |
| data | If formula_or_data is a formula, this should be either: an msd_data object, a combined dataframe, or NULL (if using observed/unobserved). |
| observed | If using formula with separate dataframes, the observed data. |
| unobserved | If using formula with separate dataframes, the unobserved data. |
| n_folds | Number of folds for cross-fitting (default 2) |
| conf_level | Confidence level for the confidence interval (default 0.95) |
| seed | Random seed for fold splitting (optional) |

Details

The PPI++ estimator uses a single tuning parameter lambda across both arms:

$$\hat{\mu}_d^{PPI}(\lambda) = \bar{Y}_{\mathcal{O}_a} + \lambda(\bar{S}_{\mathcal{U}_a}^{(d)} - \bar{S}_{\mathcal{O}_a}^{(d)})$$

The tuning parameter is estimated via cross-fitting:

1. Split labeled data into K folds
2. For each fold k, estimate lambda on the opposite folds
3. Compute the fold-k estimate using the estimated lambda
4. Average across folds

Lambda is chosen to minimize the combined variance across arms.

Value

An msd_result object containing:

| | |
|--------------------|---|
| estimate | Point estimate of the ATE |
| variance | Estimated variance (delta-method) |
| se | Standard error |
| ci_lower, ci_upper | Confidence interval bounds |
| method | Name of the estimation method |
| lambda | Estimated tuning parameter (single value) |

Note

PPI++ uses a single lambda across arms, unlike D-T which uses arm-specific tuning parameters. For arm-specific tuning, use `msd_dt`.

Examples

```

# Create sample data
set.seed(123)
n <- 100
obs_df <- data.frame(
  Y = rnorm(n),
  S0 = rnorm(n, 0, 0.5),
  S1 = rnorm(n, 0.2, 0.5),
  D = rep(c(1, 0), each = n/2)
)
obs_df$Y <- obs_df$Y + 0.3 * obs_df$D
obs_df$S1[obs_df$D == 1] <- obs_df$S1[obs_df$D == 1] + 0.5 * obs_df$Y[obs_df$D == 1]
obs_df$S0[obs_df$D == 0] <- obs_df$S0[obs_df$D == 0] + 0.5 * obs_df$Y[obs_df$D == 0]

unobs_df <- data.frame(
  S0 = rnorm(200, 0, 0.5),
  S1 = rnorm(200, 0.2, 0.5),
  D = rep(c(1, 0), each = 100)
)

msd <- msd_data(observed = obs_df, unobserved = unobs_df)
result <- msd_ppi(msd)

# Using formula interface
result2 <- msd_ppi(Y ~ D | S1 + S0, observed = obs_df, unobserved = unobs_df)

```

Description

Find optimal budget allocation between human observations and predictions. Optimal Design Selection

Determines the optimal allocation of budget between human observations and LLM predictions, and recommends the best estimator for the given pilot data.

Usage

```

optimal_design(
  pilot_data,
  budget,
  cost_human,
  cost_prediction,
  treatment_prob = 0.5,
  estimators = "all",
  min_observed = 20,
  n_grid = 100
)

```

Arguments

| | |
|-----------------|--|
| pilot_data | An msd_data object from a pilot study |
| budget | Total budget available (in dollars) |
| cost_human | Cost per human observation (in dollars) |
| cost_prediction | Cost per LLM prediction (in dollars) |
| treatment_prob | Probability of treatment assignment (default 0.5) |
| estimators | Which estimators to consider. Either "all" or a character vector with subset of: "dim", "greg", "ppi", "dt", "dip", "dip_pp", "dt_dip" |
| min_observed | Minimum number of observed units required (default 20) |
| n_grid | Number of grid points for optimization (default 100) |

Details

The function uses grid search to find the optimal (n_O , n_U) allocation that minimizes expected variance given the budget constraint:

$$n_O \times cost_{human} + n_U \times (k \times cost_{prediction}) \leq budget$$

where k is the number of predictions per unit:

- $k = 0$ for DiM (no predictions)
- $k = 1$ for GREG, PPI++, D-T (one prediction per arm)
- $k = 2$ for DiP, DiP++, D-T DiP (both $S^{(0)}$ and $S^{(1)}$)

The expected variance is computed using population moments estimated from the pilot data.

Value

An S3 object of class "msd_design" containing:

| | |
|-------------------|---|
| optimal_n_obs | Recommended number of observed (human) units |
| optimal_n_unobs | Recommended number of unobserved (prediction) units |
| optimal_estimator | Recommended estimator |
| optimal_variance | Expected variance at the optimum |
| optimal_se | Expected standard error at the optimum |
| budget_used | Total budget used |
| all_results | Full grid search results for all estimators |

Examples

```
# Pilot study data
pilot_obs <- data.frame(
  Y = rnorm(50),
  S0 = rnorm(50),
  S1 = rnorm(50),
  D = rep(c(1, 0), each = 25)
)
pilot_unobs <- data.frame(
  S0 = rnorm(100),
  S1 = rnorm(100),
  D = rep(c(1, 0), each = 50)
)
pilot <- msd_data(observed = pilot_obs, unobserved = pilot_unobs)

# Find optimal design with $10,000 budget
design <- optimal_design(
  pilot_data = pilot,
  budget = 10000,
  cost_human = 10,      # $10 per human observation
  cost_prediction = 0.01 # $0.01 per prediction
)
print(design)
```

print.msdata *Print method for msdata*

Description

Print method for msdata

Usage

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

Arguments

x An msdata object
... Additional arguments (ignored)

Value

Invisibly returns x

print.msdesign *Print method for msdesign*

Description

Print method for msdesign

Usage

```
## S3 method for class 'msdesign'  
print(x, digits = 4, ...)
```

Arguments

| | |
|--------|---------------------------------|
| x | An msdesign object. |
| digits | Number of digits to display. |
| ... | Additional arguments (ignored). |

Value

Invisibly returns x.

print.msresult *Print method for msresult*

Description

Print method for msresult

Usage

```
## S3 method for class 'msresult'  
print(x, digits = 4, ...)
```

Arguments

| | |
|--------|--------------------------------|
| x | An msresult object |
| digits | Number of digits to display |
| ... | Additional arguments (ignored) |

Value

Invisibly returns x

print.ms_d_summary *Print method for ms_d_summary*

Description

Print method for ms_d_summary

Usage

```
## S3 method for class 'ms_d_summary'  
print(x, digits = 4, ...)
```

Arguments

| | |
|--------|---------------------------------|
| x | An ms_d_summary object. |
| digits | Number of digits to display. |
| ... | Additional arguments (ignored). |

Value

Invisibly returns x.

print.summary.ms_d_result
Print method for summary.ms_d_result

Description

Produces nicely formatted output similar to summary.lm, designed for applied social scientists.

Usage

```
## S3 method for class 'summary.ms_d_result'  
print(x, digits = 4, ...)
```

Arguments

| | |
|--------|---|
| x | A summary.ms_d_result object |
| digits | Number of significant digits to display (default 4) |
| ... | Additional arguments (ignored) |

Value

Invisibly returns x

summary.ms_d_design *Summary method for ms_d_design*

Description

Summary method for ms_d_design

Usage

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

Arguments

object An ms_d_design object.
... Additional arguments passed to the print method.

Value

Invisibly returns object.

summary.ms_d_result *Summary method for ms_d_result*

Description

Produces a detailed summary of the MSD estimation results, including the coefficient table with z-statistics and p-values, sample size information, and interpretation guidance.

Usage

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

Arguments

object An ms_d_result object
... Additional arguments (ignored)

Value

A summary.ms_d_result object (invisibly when printed)

Examples

```
obs_df <- data.frame(  
  Y = rnorm(100), S0 = rnorm(100), S1 = rnorm(100),  
  D = rep(c(1, 0), each = 50)  
)  
unobs_df <- data.frame(  
  S0 = rnorm(200), S1 = rnorm(200), D = rep(c(1, 0), each = 100)  
)  
msd <- msd_data(observed = obs_df, unobserved = unobs_df)  
result <- msd_dt_dip(msd, seed = 1)  
summary(result)
```

Index

`bootstrap_variance`, [2](#)
`compare_variance_methods`, [4](#)
`estimate_all`, [5](#)
`msd_data`, [2](#), [5](#), [9](#), [10](#), [12](#), [14](#), [16](#), [18](#), [20](#)
`msd_dim`, [8](#)
`msd_dip`, [10](#)
`msd_dip_pp`, [12](#)
`msd_dt`, [13](#), [20](#)
`msd_dt_dip`, [13](#), [15](#)
`msd_greg`, [17](#)
`msd_ppi`, [17](#), [19](#)
`optimal_design`, [21](#)
`print.msd_data`, [23](#)
`print.msd_design`, [24](#)
`print.msd_result`, [24](#)
`print.msd_summary`, [25](#)
`print.summary.msd_result`, [25](#)
`summary.msd_design`, [26](#)
`summary.msd_result`, [26](#)