Package 'smoothedIPW'

May 9, 2026

Title Time-Smoothed Inverse Probability Weighting for Repeatedly Measured Outcomes

Version 0.1.0

Description Implements several methods to estimate effects of generalized time-varying treatment strategies on the mean of an outcome at one or more selected follow-up times of interest. Specifically, the package implements the time-smoothed inverse probability weighted estimators described in McGrath et al. (2025) <doi:10.48550/arXiv.2509.13971>. Outcomes may be repeatedly, non-monotonically, informatively, and sparsely measured in the data source. The package also supports settings where outcomes are truncated by death, i.e. some individuals die during follow-up which renders the outcome of interest undefined at the follow-up time of interest.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports data.table, progress

Depends R (>= 2.10)

LazyData true

BugReports https://github.com/stmcg/smoothedIPW/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Sean McGrath [aut, cre] (ORCID: <https://orcid.org/0000-0002-7281-3516>), Takuya Kawahara [aut], Jessica Young [aut] (ORCID: <https://orcid.org/0000-0002-2758-6932>)

Maintainer Sean McGrath <sean.mcgrath514@gmail.com>

Repository CRAN

Date/Publication 2026-01-09 15:00:02 UTC

Contents

`data_null` *Example dataset with null treatment effects*

Description

A dataset consisting of 25,000 observations on 1,000 individuals over 25 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
data_null
```

Format

A data table with 25,000 rows and 7 variables:

time Time index.

id Unique identifier for each individual.

L Binary time-varying covariate.

Z Medication initiated at baseline.

A Binary indicator of adhering to medication initiated at baseline.

R Indicator if the outcome of interest is measured.

Y Continuous outcome of interest.

data_null_deaths *Example dataset with null treatment effects and deaths (continuous outcome)*

Description

A dataset consisting of 21,713 observations on 1,000 individuals over 25 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
data_null_deaths
```

Format

A data table with 21,713 rows and 8 variables:

time Time index.

id Unique identifier for each individual.

L Binary time-varying covariate.

Z Medication initiated at baseline.

A Binary indicator of adhering to medication initiated at baseline.

R Indicator if the outcome of interest is measured.

Y Continuous outcome of interest.

D Indicator if death occurred.

data_null_deaths_binary *Example dataset with null treatment effects and deaths (binary outcome)*

Description

A dataset consisting of 21,674 observations on 1,000 individuals over 25 time points. Each row in the dataset corresponds to the record of one individual at one time point.

Usage

```
data_null_deaths_binary
```

Format

A data table with 21,674 rows and 8 variables:

time Time index.

id Unique identifier for each individual.

L Binary time-varying covariate.

Z Medication initiated at baseline.

A Binary indicator of adhering to medication initiated at baseline.

R Indicator if the outcome of interest is measured.

Y Binary outcome of interest.

D Indicator if death occured.

---

get_CI *Bootstrap-based confidence intervals*

Description

This function applies nonparametric bootstrap to construct confidence intervals around the counterfactual mean/probability estimates obtained by `ipw`.

Usage

```
get_CI(
  ipw_res,
  data,
  n_boot,
  conf_level = 0.95,
  reference_z_value,
  contrast_type = "difference",
  show_progress = TRUE
)
```

Arguments

`ipw_res`	Output from the `ipw` function.
`data`	Data table containing the observed data
`n_boot`	Numeric scalar specifying the number of bootstrap replicates to use
`conf_level`	Numeric scalar specifying the confidence level for the confidence intervals. The default is `0.95`.
`reference_z_value`	Scalar specifying the value of z considered as the reference level when forming contrasts. See also argument `contrast_type`.
`contrast_type`	Character string specifying the type of contrast. The options are `"difference"` (for the difference of means/probabilities) and `"ratio"` (for the ratio of means/probabilities).
`show_progress`	Logical scalar specifying whether to show a progress bar.

Details

This function applies nonparametric bootstrap resampling to construct confidence intervals around the counterfactual mean/probability estimates obtained by `ipw`. Bootstrap confidence intervals are constructed by resampling individuals (with replacement) from the original data set, applying the `ipw` function to each bootstrap sample, and computing percentile-based confidence intervals from the distribution of bootstrap estimates.

Value

An object of class "ipw_ci". This object is a list that includes the following components:

`res_boot` A list where each component corresponds to a different medication z level. Each component of the list is a data frame containing the estimates and confidence intervals for the counterfactual outcome mean/probability under the treatment regime indexed by z.

`res_boot_contrast` A list where each component corresponds to a different medication z level. Each component of the list is a data frame containing the estimates and confidence intervals for the contrast (difference or ratio) counterfactual outcome mean/probability under the treatment regime indexed by z compared to the counterfactual outcome mean/probability under the treatment regime indexed by the reference value.

`res_boot_all` A three dimensional array containing all the bootstrap replicates. The first dimension corresponds to the bootstrap replicate; The second dimension corresponds to the time interval; The third dimension corresponds to the medication z level.

`outcome_type` Character string indicating whether the outcome is "continuous" or "binary".

`outcome_times` Numeric vector of outcome times.

`n_boot` Number of bootstrap replicates used.

`conf_level` Confidence level used.

`reference_z_value` Reference value of Z used for contrasts.

`contrast_type` Type of contrast ("difference" or "ratio").

Examples

```
set.seed(1234)
data_null_processed <- prep_data(data = data_null, grace_period_length = 2,
                                 baseline_vars = 'L')
res_est <- ipw(data = data_null_processed,
               time_smoothed = TRUE,
               outcome_times = c(6, 12, 18, 24),
               A_model = A ~ L + Z,
               R_model_numerator = R ~ L_baseline + Z,
               R_model_denominator = R ~ L + A + Z,
               Y_model = Y ~ L_baseline * (time + Z))
res_ci <- get_CI(ipw_res = res_est, data = data_null_processed, n_boot = 10)
res_ci
```

ipw *Time-smoothed inverse probability weighting*

Description

This function applies the time-smoothed inverse probability weighted (IPW) approach described by McGrath et al. (2025) to estimate effects of generalized time-varying treatment strategies on the mean of an outcome at one or more selected follow-up times of interest. Binary and continuous outcomes are supported.

Usage

```
ipw(
  data,
  time_smoothed = TRUE,
  smoothing_method = "nonstacked",
  outcome_times,
  A_model,
  R_model_numerator = NULL,
  R_model_denominator,
  Y_model,
  truncation_percentile = NULL,
  include_baseline_outcome,
  return_model_fits = TRUE,
  return_weights = TRUE,
  trim_returned_models = FALSE
)
```

Arguments

`data`	Data table (or data frame) containing the observed data. See "Details".
`time_smoothed`	Logical scalar specifying whether the time-smoothed or non-smoothed IPW method is applied. The default is `TRUE`, i.e., the time-smoothed IPW method.
`smoothing_method`	Character string specifying the time-smoothed IPW method when there are deaths present. The options include `"nonstacked"` and `"stacked"`. The default is `"nonstacked"`.
`outcome_times`	Numeric vector specifying the follow-up time(s) of interest for the counterfactual outcome mean/probability
`A_model`	Model statement for the treatment variable

`R_model_numerator`
: (Optional) Model statement for the indicator variable for the measurement of the outcome variable, used in the numerator of the IP weights. The default is `NULL`, i.e., a numerator of 1 is used in the IP weights.

`R_model_denominator`
: Model statement for the indicator variable for the measurement of the outcome variable, used in the denominator of the IP weights

`Y_model`
: Model statement for the outcome variable

`truncation_percentile`
: Numerical scalar specifying the percentile by which to truncate the IP weights. The default is `NULL`, i.e., no truncation.

`include_baseline_outcome`
: Logical scalar indicating whether to include the time interval indexed by 0 in fitting the time-smoothed outcome model and outcome measurement models. By default, this argument is set to `TRUE` if `data` has any non-missing outcome values in the time interval indexed by 0 and is otherwise set to `FALSE`.

`return_model_fits`
: Logical scalar specifying whether to include the fitted models in the output. The default is `TRUE`.

`return_weights`
: Logical scalar specifying whether to return the estimated inverse probability weights. The default is `TRUE`.

`trim_returned_models`
: Logical scalar specifying whether to only return the estimated coefficients (and corresponding standard errors, z scores, and p-values) of the fitted models (e.g., treatment model) rather than the full fitted model objects. This reduces the size of the object returned by the `ipw` function when `return_model_fits` is set to `TRUE`, especially when the observed data set is large. By default, this argument is set to `FALSE`.

Details

Treatment strategies

Users can estimate effects of treatment strategies with the following components:

- Initiate treatment z at baseline
- Follow a user-specified time-varying adherence protocol for treatment z
- Ensure an outcome measurement at the follow-up time of interest.

The time-varying adherence protocol is specified by indicating in `data` when an individual deviates from their adherence protocol. The function `prep_data` facilitates this step. See also "Formatting `data`".

Formatting `data`

The input data set `data` must be a data table (or data frame) in a "long" format, where each row represents one time interval for one individual. The data frame should contain the following columns:

- `id`: A unique identifier for each participant.

- time: The follow-up time index, starting from 0 and increasing in increments of 1 in consecutive rows.
- Covariate columns: One or more columns for baseline and time-varying covariates.
- Z: The treatment initiated at baseline.
- A: An indicator for adherence to the treatment protocol at each time point.
- R: An indicator of whether the outcome was measured at that time point (1 for measured, 0 for not measured/censored).
- Y: The outcome variable, which can be binary or continuous.

To specify the intervention, the data set should additionally have the following columns:

- C_artificial: An indicator specifying when an individual should be artificially censored from the data due to violating the adherence protocol.
- A_model_eligible: An indicator specifying which records should be used for fitting the treatment adherence model.

The prep_data function facilitates adding these columns to the data set. Users may optionally include the following column for fitting the outcome measurement model:

- R_model_denominator_eligible: An indicator specifying which records should be used for fitting the outcome measurement model R_model_denominator_eligible.

Otherwise, the R_model_denominator_eligible is fit on all records on the artificially censored data set.

Specifying the models

Users must specify model statements for the treatment (A_model), outcome measurement (R_model_numerator and R_model_denominator), and outcome variable (Y_model). The package uses pooled-over-time generalized linear models that are fit over the relevant time points (see "Formatting data"), where logistic regression is used for binary variables and linear regression is used for continuous variables.

For stabilized weights, the outcome measurement model R_model_numerator should **only** include baseline covariates, treatment initiated Z, and time as predictors. It must not include time-varying covariates as predictors. The outcome model Y_model should also only depend on baseline covariates, treatment initiated Z, and time (if using time smoothing).

A note on the outcome definition at baseline

In some settings, the outcome may not be defined in the baseline time interval. The ipw function can accommodate such settings in two ways:

1. Users can set a value of NA in the column Y in the input data set data in rows corresponding to time 0. In this case, users should ensure that include_baseline_outcome is set to FALSE.
2. Users can specify the value of Y_{t+1} (rather than Y_t) in the column Y in the input data set data in rows corresponding to time t. That is, the value supplied for Y in the input data set data at time 0 is Y_1. In this case, users should ensure that include_baseline_outcome is set to TRUE. Users should also set outcome_times accordingly.

Note that these two approaches involve different assumptions. For example, the first approach allows the outcome at time t to depend on time-varying covariates up to and including time t, whereas the second approach only allows the outcome at time t to depend on covariates up to and including time $t-1$.

Value

An object of class "ipw". This object is a list that includes the following components:

`est` A data frame containing the counterfactual mean/probability estimates for each medication at each time interval.

`model_fits` A list containing the fitted models for the treatment, outcome measurement, and outcome (if `return_model_fits` is set to `TRUE`). If the nonstacked time-smoothed approach is used, the ith element in `model_fits` is a list of fitted models for the ith outcome time in `outcome_times`. If the stacked time-smoothed approach is used, the ith element in `model_fits` is a list of fitted models for the outcome time $i + 1$ in the data set `data`. The last element in `model_fits` contains the fitted outcome model.

`data_weights` (A list containing) the artificially censored data set with columns for the estimated weights. The column `"weights"` contains the (final) inverse probability weight, and the columns `"weights_A"` and `"weights_R"` contain the inverse probability weights for treatment and outcome measurement, respectively. If no deaths are present in the data, this object will be a data frame. If deaths are present in the data and either the non-smoothed IPW method is applied or the time-smoothed non-stacked IPW method is applied, this object will be a list of length `length(outcome_times)` where each element corresponds to the artificially censored data set for each outcome time in `outcome_times`. If deaths are present in the data and the time-smoothed stacked IPW method is applied, this object will be a data frame with the stacked, artificially censored data.

`args` A list containing the arguments supplied to `ipw`, except the observed data set.

References

McGrath S, Kawahara T, Petimar J, Rifas-Shiman SL, Díaz I, Block JP, Young JG. (2025). Time-smoothed inverse probability weighted estimation of effects of generalized time-varying treatment strategies on repeated outcomes truncated by death. arXiv e-prints arXiv:2509.13971.

Examples

```
## Time-smoothed IPW without deaths (continuous outcome)
data_null_processed <- prep_data(data = data_null, grace_period_length = 2,
                                 baseline_vars = 'L')
res <- ipw(data = data_null_processed,
           time_smoothed = TRUE,
           outcome_times = c(6, 12, 18, 24),
           A_model = A ~ L + Z,
           R_model_numerator = R ~ L_baseline + Z,
           R_model_denominator = R ~ L + A + Z,
           Y_model = Y ~ L_baseline * (time + Z))
res

## Time-smoothed IPW with deaths, nonstacked smoothing method (continuous outcome)
data_null_deaths_processed <- prep_data(data = data_null_deaths, grace_period_length = 2,
                                        baseline_vars = 'L')
res <- ipw(data = data_null_deaths_processed,
```

```
            time_smoothed = TRUE,
            smoothing_method = 'nonstacked',
            outcome_times = c(6, 12, 18, 24),
            A_model = A ~ L + Z,
            R_model_numerator = R ~ L_baseline + Z,
            R_model_denominator = R ~ L + A + Z,
            Y_model = Y ~ L_baseline * (time + Z))
res

## Time-smoothed IPW with deaths, stacked smoothing method (binary outcome)

data_null_deaths_binary_processed <- prep_data(data = data_null_deaths_binary,
                                               grace_period_length = 2,
                                               baseline_vars = 'L')
res <- ipw(data = data_null_deaths_binary_processed,
            time_smoothed = TRUE,
            smoothing_method = 'stacked',
            outcome_times = c(6, 12, 18, 24),
            A_model = A ~ L + Z,
            R_model_numerator = R ~ L_baseline + Z,
            R_model_denominator = R ~ L + A + Z,
            Y_model = Y ~ L_baseline * (time + Z))
res$est
```

`prep_data` *Prepare data set for inverse probability weighting*

Description

This function adds columns to the input data set to assist with inverse probability weighting. See details.

Usage

```
prep_data(
  data,
  grace_period_length = 0,
  baseline_vars = NULL,
  lag_vars = NULL,
  n_lags = 1
)
```

Arguments

`data`	Data frame containing the observed data
`grace_period_length`	Numeric scalar indicating the length of the grace period, if applicable. The default is `0`, indicating no grace period.

baseline_vars	Vector of character strings specifying the names of the baseline covariates that should be added to the observed data.
lag_vars	Vector of character strings specifying the names of the covariates whose lags should be added as columns to the observed data. The number of lags is controlled by the n_lags argument.
n_lags	Numeric scalar specifying the number of lags to use when computing the lagged values of lag_vars. Additional columns will be created for 1, ..., n_lags lags of the variables specified in lag_vars.

Details

This function performs the following tasks:

- Adds a column C_artificial which indicates when an individual should be artificially censored from the data when applying inverse probability weighting.
- Adds a column A_model_eligible which indicates what records should be used for fitting the treatment adherence model.
- If baseline_vars is supplied, it adds columns corresponding to the baseline value of these variables. These columns have the name _baseline appended to them.
- If lag_vars is supplied, it adds columns corresponding to the lagged value of these variables. For each of these variables, additional columns will be created for 1, ..., n_lags lags of the variable.

Value

A data table containing the observed data with the additional columns.

Examples

```
data_null_processed <- prep_data(data = data_null, grace_period_length = 2,
                                 baseline_vars = 'L')
```

---

print.ipw *Print method for "ipw" objects*

Description

Print method for objects of class "ipw".

Usage

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

Arguments

x	Object of class "ipw".
...	Other arguments.

Value

No value is returned.

See Also

ipw

Examples

```
data_null_processed <- prep_data(data = data_null, grace_period_length = 2,
                                 baseline_vars = 'L')
res <- ipw(data = data_null_processed,
           time_smoothed = TRUE,
           outcome_times = c(6, 12, 18, 24),
           A_model = A ~ L + Z,
           R_model_numerator = R ~ L_baseline + Z,
           R_model_denominator = R ~ L + A + Z,
           Y_model = Y ~ L_baseline * (time + Z))
print(res)
```

---

print.ipw_ci *Print method for "ipw_ci" objects*

Description

Print method for objects of class "ipw_ci".

Usage

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

Arguments

x	Object of class "ipw_ci".
...	Other arguments.

Value

No value is returned.

See Also

get_CI

Examples

```
set.seed(1234)
data_null_processed <- prep_data(data = data_null, grace_period_length = 2,
                                 baseline_vars = 'L')
res_est <- ipw(data = data_null_processed,
               time_smoothed = TRUE,
               outcome_times = c(6, 12, 18, 24),
               A_model = A ~ L + Z,
               R_model_numerator = R ~ L_baseline + Z,
               R_model_denominator = R ~ L + A + Z,
               Y_model = Y ~ L_baseline * (time + Z))
res_ci <- get_CI(ipw_res = res_est, data = data_null_processed, n_boot = 10)
print(res_ci)
```

Index