

Package: SensIAT (via r-universe)

October 17, 2026

Title Sensitivity Analysis for Irregular Assessment Times

Version 0.3.0.9000

Author Andrew Redd [aut, cre], Yujing Gao [aut], Shu Yang [aut],
Bonnie Smith [aut], Ravi Varadhan [aut], Agatha Mallett [ctb,
ctr], Daniel Scharfstein [pdr, aut], University of Utah [cph]

Maintainer Andrew Redd <andrew.redd@hsc.utah.edu>

Description Sensitivity analysis for trials with irregular and
informative assessment times, based on a new influence
function-based, augmented inverse intensity-weighted estimator.

Language en-US

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports assertthat, BB, dplyr, generics, ggplot2, glue, KernSmooth,
lifecycle, MASS, MAVE, methods, orthogonalSplineBasis, pracma,
purrr, Rcpp (>= 1.0.12), rlang, splines, stats, survival,
tibble, tidyr, utils

Suggests bench, covr, dfoptim, furrr, future, inline, knitr,
ManifoldOptim, metR, numDeriv, progress, quarto, rmarkdown,
spelling, statmod, testthat (>= 3.0.0), tidyverse

Config/testthat/edition 3

Depends R (>= 4.4.0)

LazyData true

LinkingTo Rcpp

SystemRequirements C++17

URL <https://github.com/UofUEpiBio/SensIAT>,
<https://uofuepibio.github.io/SensIAT/>

BugReports <https://github.com/UofUEpiBio/SensIAT/issues>

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev

Repository <https://uofuepibio.r-universe.dev>

Date/Publication 2026-07-01 05:03:32 UTC

RemoteUrl <https://github.com/UofUEpiBio/SensIAT>

RemoteRef HEAD

RemoteSha 6fb11997f2d674fb4c13351ed708ffc2b126f35f

Contents

autoplot.SensIAT_fulldata_jackknife_results	3
autoplot.SensIAT_fulldata_model	4
autoplot.SensIAT_within_group_model	5
autoplot.SensIAT_withingroup_bootstrap_results	6
autoplot.SensIAT_withingroup_jackknife_results	6
benchmark_term2_methods	7
compute_influence_terms	9
compute_SensIAT_expected_values	11
fit_marginal_model	13
fit_SensIAT_fulldata_model	14
fit_SensIAT_marginal_mean_model	17
fit_SensIAT_marginal_mean_model_generalized	19
fit_SensIAT_single_index_fixed_coef_model	22
fit_SensIAT_single_index_norm1coef_model	24
get_bootstrap_coeffs	25
jackknife	26
make_parametric_intensity_simulator	27
make_parametric_single_index_simulator	28
make_single_index_simulator	29
parametric_bootstrap	29
parametric_bootstrap_within_group	30
predict.SensIAT_fulldata_model	32
predict.SensIAT_withingroup_bootstrap_results	33
prepare_SensIAT_data	34
sample_parametric_coeffs	35
SensIAT_example_data	36
simulate_SensIAT_data	37
simulate_SensIAT_two_groups	39

Index	43
--------------	-----------

```
autoplot.SensIAT_fulldata_jackknife_results
      Plot      for      Estimated      Treatment      Effect      for
      SensIAT_fulldata_jackknife_results Objects
```

Description

The horizontal and vertical axes represent the sensitivity parameter **alpha** for the control and treatment groups, respectively. The plot shows at each combination of **alpha** values zero if the **level** confidence interval contains zero, otherwise the bound of the confidence interval that is closest to zero.

Usage

```
## S3 method for class 'SensIAT_fulldata_jackknife_results'
autoplot(object, level = 0.05, ..., include.rugs = NA)
```

Arguments

object	A <code>SensIAT_fulldata_jackknife_results</code> object.
level	Significance level of the confidence interval.
...	Additional arguments passed to <code>predict</code> .
include.rugs	If <code>TRUE</code> , adds rugs to the plot. If <code>FALSE</code> , no rugs are added. When <code>NA</code> , rugs are added only if the number of distinct values of alpha_control and alpha_treatment is less than or equal to 10.

Examples

```
# Note: fitting the jackknife is computationally expensive,
#       so this example is here for reference.
## Not run:
full.object <-
  fit_SensIAT_fulldata_model(
    data = SensIAT_example_fulldata,
    trt = Treatment_group == "treatment",
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    knots = c(60, 260, 460),
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6)
  )
jk.full.model <- jackknife(full.object, time = 180)
ggplot2::autoplot(jk.full.model)

## End(Not run)
```

```
autoplot.SensIAT_fulldata_model
      Plot      for      Estimated      Treatment      Effect      for
SensIAT_fulldata_model Objects
```

Description

The horizontal and vertical axes represent the sensitivity parameter **alpha** for the control and treatment groups, respectively. The contour plot shows the estimated treatment effect at each combination of **alpha** values.

Usage

```
## S3 method for class 'SensIAT_fulldata_model'
autoplot(object, time, include.rugs = NA, ...)
```

Arguments

<code>object</code>	A <code>SensIAT_fulldata_model</code> object.
<code>time</code>	Time at which to plot the estimates.
<code>include.rugs</code>	If <code>TRUE</code> , adds rugs indicating the locations where the sensitivity was evaluated to the plot. If <code>FALSE</code> , no rugs are added. If <code>NA</code> , rugs are added only if the number of distinct values of <code>alpha_control</code> and <code>alpha_treatment</code> is less than or equal to 10.
<code>...</code>	Additional arguments passed to <code>predict</code> .

Value

A `ggplot2` object.

Examples

```
full.object <-
  fit_SensIAT_fulldata_model(
    data = SensIAT_example_fulldata,
    trt = Treatment_group == "treatment",
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    knots = c(60, 260, 460),
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6)
  )
ggplot2::autoplot(full.object, time = 180)
```

autoplot.SensIAT_within_group_model

Plot a SensIAT_within_group_model Object

Description

This creates a line plot for a `SensIAT_within_group_model` object. The horizontal axis represents time, and the vertical axis represents the expected marginal outcome given the sensitivity parameter `alpha`.

Usage

```
## S3 method for class 'SensIAT_within_group_model'
autoplot(object, ...)
```

Arguments

<code>object</code>	A <code>SensIAT_within_group_model</code> object.
<code>...</code>	currently ignored

Value

A `ggplot2` object.

Examples

Note: example takes a few seconds to run.

```
object <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_bandwidth_model,
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    knots = c(60, 260, 460),
    End = 830,
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6),
    intensity.args = list(bandwidth = 30)
  )
ggplot2::autoplot(object) +
  # Title not included
  ggplot2::ggtitle("SensIAT within group model") +
  # Nor are bounds on reasonable values of alpha
  ggplot2::geom_hline(yintercept = c(1.2, 3), linetype = "dotted", linewidth = 1.5)
```

```
autoplot.SensIAT_withingroup_bootstrap_results
```

Plot Bootstrap Marginal Mean Curves with Confidence Bands

Description

Plot Bootstrap Marginal Mean Curves with Confidence Bands

Usage

```
## S3 method for class 'SensIAT_withingroup_bootstrap_results'
autoplot(object, time = NULL, level = 0.95, ...)
```

Arguments

<code>object</code>	A <code>SensIAT_withingroup_bootstrap_results</code> object.
<code>time</code>	Optional vector of times. If <code>NULL</code> , uses an internal regular grid.
<code>level</code>	Confidence level used for bootstrap intervals.
<code>...</code>	Ignored.

Value

A `ggplot2` object.

```
autoplot.SensIAT_withingroup_jackknife_results
```

*Plot Estimates at Given Times for
SensIAT_withingroup_jackknife_results Objects*

Description

Horizontal axis represents time, and the vertical axis represents the outcome from the model. Point plotted is the mean estimate, and the error bars show the `level` confidence interval using the variance estimated from the jackknife.

Usage

```
## S3 method for class 'SensIAT_withingroup_jackknife_results'
autoplot(object, level = 0.95, width = NULL, ...)
```

Arguments

<code>object</code>	A <code>SensIAT_withingroup_jackknife_results</code> object produced from <code>SensIAT_jackknife</code> .
<code>level</code>	Significance level of the confidence interval.
<code>width</code>	Width of the dodge for position, default is half the minimum distance between time evaluation points.
<code>...</code>	Ignored.

Value

A ggplot2 object.

Examples

```
# Note: fitting the jackknife is computationally expensive,
#       so this example is here for reference.
## Not run:
fitted <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6),
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    intensity.args = list(bandwidth = 30),
    knots = c(60, 260, 460),
    End = 830
  )
jackknife.estimates <- SensIAT_jackknife(fitted, time = c(90, 180, 270, 360, 450))
ggplot2::autoplot(jackknife.estimates)

## End(Not run)
```

benchmark_term2_methods

Benchmark Term2 Integration Methods

Description

Efficiently compares different term2 integration methods by isolating just the term2 computation from the full model fitting. This is useful for performance analysis and method selection without the overhead of repeated model fitting.

Usage

```
benchmark_term2_methods(
  data,
  id,
  time,
  outcome.model,
  knots,
  spline_degree = 3L,
  alpha = 0,
  impute_data,
  link = c("identity", "log"),
  methods = c("fast", "original", "fixed_grid", "seeded_adaptive"),
```

```

    grid_sizes = c(50, 100, 200),
    n_patients = NULL,
    n_iterations = 3,
    reference_method = "fast",
    seed = 42
  )

```

Arguments

<code>data</code>	Data frame with longitudinal observations
<code>id</code>	Unquoted column name for subject identifier
<code>time</code>	Numeric vector of observation times
<code>outcome.model</code>	A fitted outcome model (e.g., from <code>fit_SensIAT_single_index_fixed_coef_model()</code>)
<code>knots</code>	Knots for marginal mean model spline basis
<code>spline_degree</code>	Degree of B-spline basis (default: 3)
<code>alpha</code>	Numeric vector of sensitivity parameters to test
<code>impute_data</code>	Function to impute data at arbitrary times: <code>function(t, patient_data) -> data.frame</code>
<code>link</code>	Link function: "identity" or "log"
<code>methods</code>	Character vector of methods to benchmark. Options: "fast", "original", "fixed_grid", "seeded_adaptive", "gauss_legendre"
<code>grid_sizes</code>	For grid-based methods, vector of grid sizes to test (default: <code>c(50, 100, 200)</code>)
<code>n_patients</code>	Number of patients to use for benchmark (NULL = all patients)
<code>n_iterations</code>	Number of timing iterations per method
<code>reference_method</code>	Method to use as accuracy reference (default: "fast")
<code>seed</code>	Random seed for reproducibility

Value

A list with components:

timing Data frame with timing results per method
accuracy Data frame with accuracy metrics vs reference
reference_results List of reference results per patient
setup_info List with setup parameters and dimensions

Examples

```

## Not run:
# Setup data with lag variables
data_with_lags <- SensIAT_example_data |>
  dplyr::group_by(Subject_ID) |>
  dplyr::mutate(
    ..prev_outcome.. = dplyr::lag(Outcome, order_by = Time),

```

```

    ..prev_time.. = dplyr::lag(Time, default = 0, order_by = Time),
    ..delta_time.. = Time - dplyr::lag(Time, order_by = Time)
  ) |>
  dplyr::ungroup()

# Fit outcome model
outcome.model <- fit_SensIAT_single_index_fixed_coef_model(
  Outcome ~ splines::ns(..prev_outcome.., df = 3) + ..delta_time.. - 1,
  id = Subject_ID,
  data = data_with_lags |> dplyr::filter(Time > 0)
)

# Imputation function
impute_fn <- function(t, df) {
  extrapolate_from_last_observation(t, df, "Time",
    slopes = c("..delta_time.." = 1))
}

# Run benchmark
results <- benchmark_term2_methods(
  data = data_with_lags,
  id = Subject_ID,
  time = data_with_lags$Time,
  outcome.model = outcome.model,
  knots = c(60, 260, 460),
  alpha = c(-0.1, 0, 0.1),
  impute_data = impute_fn,
  methods = c("fast", "fixed_grid", "seeded_adaptive"),
  grid_sizes = c(50, 100),
  n_patients = 10
)

# View timing results
print(results$timing)

## End(Not run)

```

compute_influence_terms

Compute Influence Terms

Description

This function computes the influence terms for the marginal outcome model sensitivity analysis. It is a generic function that can handle different types of outcome models.

Usage

```
compute_influence_terms(outcome.model, intensity.model, alpha, data, ...)
```

```

## Default S3 method:
compute_influence_terms(
  outcome.model,
  intensity.model,
  alpha,
  data,
  id,
  base,
  ...
)

## S3 method for class 'glm'
compute_influence_terms(
  outcome.model,
  intensity.model,
  alpha,
  data,
  base,
  tolerance = .Machine$double.eps^(1/3),
  na.action = na.fail,
  id = NULL,
  time = NULL,
  ...
)

## S3 method for class '`SensIAT::Single-index-outcome-model`'
compute_influence_terms(
  outcome.model,
  intensity.model,
  alpha,
  data,
  base,
  tolerance = .Machine$double.eps^(1/3),
  na.action = na.fail,
  id = NULL,
  time = NULL,
  ...
)

```

Arguments

<code>outcome.model</code>	The outcome model fitted to the data.
<code>intensity.model</code>	The intensity model fitted to the data.
<code>alpha</code>	A numeric vector representing the sensitivity parameter.
<code>data</code>	A data frame containing the observations.
<code>...</code>	Additional arguments passed to the method.
<code>id</code>	A variable representing the patient identifier.

base	A spline basis object.
tolerance	Numeric value indicating the tolerance for integration, default is <code>.Machine\$double.eps^(1/3)</code> .
na.action	Function to handle missing values, default is <code>na.fail</code> .
time	Variable indicating the time variable in the data, by Default will be extracted from the intensity model response.

Methods (by class)

- `compute_influence_terms(default)`: Generic method, which throws a not implemented error.
- `compute_influence_terms(glm)`: Method for Generalized Linear Models (GLM).
- `compute_influence_terms(`SensIAT::Single-index-outcome-model`)`: Optimized method for the single index model.

`compute_SensIAT_expected_values`

Compute Conditional Expected Values based on Outcome Model

Description

Compute Conditional Expected Values based on Outcome Model

Usage

```
compute_SensIAT_expected_values(
  model,
  alpha = 0,
  new.data = model.frame(model),
  ...
)

## S3 method for class 'lm'
compute_SensIAT_expected_values(model, alpha, new.data, ...)

## S3 method for class 'glm'
compute_SensIAT_expected_values(
  model,
  alpha,
  new.data,
  ...,
  y.max = NULL,
  eps = .Machine$double.eps
)

## S3 method for class 'negbin'
compute_SensIAT_expected_values(
```

```

model,
alpha,
new.data,
...,
y.max = NULL,
eps = .Machine$double.eps^(1/4)
)

```

Arguments

<code>model</code>	An object representing the output of the outcome model.
<code>alpha</code>	The sensitivity parameter
<code>new.data</code>	Data to compute conditional means for, defaults to the model frame for the fitted model.
<code>...</code>	passed onto methods.
<code>y.max</code>	The maximum value of the outcome variable for the Poisson and Negative Binomial models. If omitted it is chosen from the quantile function for the distribution at $1-\text{eps}$.
<code>eps</code>	The tolerance for the quantile function used to estimate <code>y.max</code> , default is <code>.Machine\$double.eps</code> .

Details

Compute the conditional expectations needed for predictions in the models. Two additional values/expectations are computed:

- $E \big[Y(t) \exp \{ \alpha Y(t) \} \mid A(t)=1, \bar{0}(t) \big]$, returned as `E_Yexp_alphaY`, and
- $E \big[\exp \{ \alpha Y(t) \} \mid A(t)=1, \bar{0}(t) \big]$, returned as `E_exp_alphaY`.

For the methods shown here

Value

The `new.data` frame with additional columns `alpha`, `E_Yexp_alphaY`, and `E_exp_alphaY` appended.

Methods (by class)

- `compute_SensIAT_expected_values(lm)`: (Gaussian) Linear Model method Uses closed-form analytical solutions for the conditional expectations. For Y distributed as $\text{Normal}(\mu, \sigma^2)$: $E[\exp(\alpha Y)] = \exp(\alpha \mu + \alpha^2 \sigma^2 / 2)$ and $E[Y \exp(\alpha Y)] = (\mu + \alpha \sigma^2) \exp(\alpha \mu + \alpha^2 \sigma^2 / 2)$
- `compute_SensIAT_expected_values(glm)`: Generalized Linear Model method
- `compute_SensIAT_expected_values(negbin)`: Negative Binomial Model method

Examples

```

model <- lm(mpg ~ as.factor(cyl) + disp + wt, data = mtcars)
compute_SensIAT_expected_values(model, alpha = c(-0.3, 0, 0.3), new.data = mtcars[1:5, ])
model <- glm(cyl ~ mpg + disp + wt, data = mtcars, family = poisson())
compute_SensIAT_expected_values(model, alpha = c(-0.3, 0, 0.3), new.data = mtcars[1:5, ]) |>
  dplyr::mutate("E(y|alpha)" = .data$E_Yexp_alphaY / .data$E_exp_alphaY)

```

fit_marginal_model	<i>Fit SensIAT Marginal Mean Model (Unified)</i>
--------------------	--

Description

This function fits a marginal mean model for sensitivity analysis, supporting both single-index (linear) and generalized linear outcome models.

Usage

```

fit_marginal_model(
  data,
  outcome.model,
  intensity.model,
  id,
  time,
  alpha = 0,
  knots,
  ...
)

```

Arguments

data	Data frame containing all required variables.
outcome.model	Outcome model object or formula (GLM, single-index, or LM).
intensity.model	Intensity model object (e.g., <code>coxph</code>).
id	Subject identifier variable.
time	Time variable.
alpha	Sensitivity parameter(s).
knots	Spline knot locations.
...	Additional arguments passed to underlying methods.

Value

A fitted SensIAT marginal mean model object.

```
fit_SensIAT_fulldata_model
```

Produce fitted model for group (treatment or control)

Description

Produces a fitted model that may be used to produce estimates of mean and variance for the given group.

Usage

```
fit_SensIAT_fulldata_model(data, trt, ...)
```

```
fit_SensIAT_within_group_model(
  group.data,
  outcome_modeler,
  id,
  outcome,
  time,
  knots,
  alpha = 0,
  End = NULL,
  intensity.args = list(),
  outcome.args = list(),
  influence.args = list(),
  spline.degree = 3,
  add.terminal.observations = TRUE,
  loss = c("lp_mse", "quasi-likelihood"),
  link = c("identity", "log", "logit"),
  term2_method = c("fast", "original", "fixed_grid", "seeded_adaptive", "gauss_legendre"),
  impute_data = NULL,
  progress = FALSE
)
```

Arguments

<code>data</code>	the full data set.
<code>trt</code>	an expression that determine what is treated as the treatment. Everything not treatment is considered control.
<code>...</code>	common arguments passed to <code>fit_SensIAT_within_group_model</code> .
<code>group.data</code>	The data for the group that is being analyzed. Preferably passed in as a single <code>tibble</code> that internally is subsetted/filtered as needed.
<code>outcome_modeler</code>	function for fitting the outcome model. Called with a formula, data argument and <code>outcome.args</code> list.
<code>id</code>	The variable that identifies the patient.

<code>outcome</code>	The variable that contains the outcome.
<code>time</code>	The variable that contains the time.
<code>knots</code>	knot locations for defining the spline basis.
<code>alpha</code>	The sensitivity parameter.
<code>End</code>	The end time for this data analysis, we need to set the default value as the max value of the time.
<code>intensity.args</code>	A list of optional arguments for intensity model. See the Intensity Arguments section.
<code>outcome.args</code>	parameters as needed passed into the <code>outcome_modeler</code> . One special element may be <code>'model.modifications'</code> which, if present, should be a formula that will be used to modify the outcome model per, update.formula .
<code>influence.args</code>	A list of optional arguments used when computing the influence. See the Influence Arguments section.
<code>spline.degree</code>	The degree of the spline basis.
<code>add.terminal.observations</code>	Logical indicating whether to add terminal observations to the data. If TRUE, data may not contain any NAs. if FALSE, data will be assumed to already include the terminal observations
<code>loss</code>	The loss function to use. Options are <code>"lp_mse"</code> (default) and <code>"quasi-likelihood"</code> . Only used when <code>link</code> is not <code>"identity"</code> .
<code>link</code>	The link function to use. Options are <code>"identity"</code> (default), <code>"log"</code> , and <code>"logit"</code> . When <code>"identity"</code> , the original analytic formula is used; otherwise, the generalized iterative solver is used.
<code>term2_method</code>	Method for computing term2 influence components. Options are: <code>"fast"</code> (default), <code>"original"</code> , <code>"fixed_grid"</code> , <code>"seeded_adaptive"</code> , <code>"gauss_legendre"</code> . Only used when <code>link</code> is not <code>"identity"</code> .
<code>impute_data</code>	A function that takes <code>(t, df)</code> and returns the imputed data at time <code>t</code> . If NULL (default), a standard imputation function is generated automatically.
<code>progress</code>	Logical indicating whether to print stage progress messages. Default is FALSE.

Details

This function should be agnostic to whether it is being provided a treatment or control group.

Value

a list with class `SensIAT-fulldata-fitted-model` with two components, `control` and `treatment`, each of which is an independently fitted `SensIAT-within-group-fitted-model` fit with the `fit_within_group_model` function.

A `SensIAT_within_group_model` object containing:

- **models**: List with **intensity** and **outcome** sub-models
- **data**: The transformed data used for fitting
- **variables**: List of variable names (id, outcome, time)
- **End**: The end time for the analysis
- **influence**: Influence function values by patient
- **alpha**: Sensitivity parameter values
- **coefficients**: Estimated spline coefficients for each alpha
- **coefficient.variance**: Variance of coefficients for each alpha
- **base**: The [SplineBasis](#) object
- **V_inverse**: Inverse of the Gram matrix
- **link**: The link function used
- **loss**: The loss function used
- **term2_method**: The term2 integration method used

Functions

- **fit_SensIAT_fulldata_model()**: Fit the Marginal Mean Sensitivity Analysis with Tilting Assumption for Both Treatment and Control Groups.

Intensity Arguments

The **intensity.args** list may contain the following elements:

- **model.modifications** A formula that will be used to modify the intensity model from it's default, per [update.formula](#).
- **kernel** The kernel function for the intensity model. Default is the Epanechnikov kernel.
- **bandwidth** The bandwidth for the intensity model kernel.

Influence Arguments

The **influence.args** list may contain the following elements:

- **method** The method for integrating, adaptive or fixed quadrature. Default is '**adaptive**'.
- **tolerance** The tolerance when using adaptive quadrature.
- **delta** The bin width for fixed quadrature.
- **resolution** alternative to **delta** by specifying the number of bins.
- **fix_discontinuity** Whether to account for the discontinuity in the influence at observation times.

Examples

```
# Example 1: Default identity link (original analytic solution)
model <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6),
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    End = 830,
    knots = c(60, 260, 460),
  )

# Example 2: Log link with generalized iterative solver
model_log <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    alpha = 0,
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    End = 830,
    knots = c(60, 260, 460),
    link = "log",
    loss = "lp_mse"
  )
```

```
fit_SensIAT_marginal_mean_model
  Fit the Marginal Means Model
```

Description

Fit the Marginal Means Model

Usage

```
fit_SensIAT_marginal_mean_model(
  data,
  id,
  alpha,
  knots,
  outcome.model,
  intensity.model,
  spline.degree = 3L,
```

```
    ...
  )
```

Arguments

<code>data</code>	Data for evaluation of the model. Should match the data used to fit the intensity and outcome models.
<code>id</code>	The subject identifier variable in the data. Lazy evaluation is used, so it can be a symbol or a string.
<code>alpha</code>	Sensitivity parameter, a vector of values.
<code>knots</code>	Location of spline knots. If a <code>SplineBasis</code> object is provided, it is used directly.
<code>outcome.model</code>	The observed effects model.
<code>intensity.model</code>	The assessment time intensity model.
<code>spline.degree</code>	The degree of the spline basis, default is 3 (cubic splines).
<code>...</code>	Additional arguments passed to <code>compute_influence_terms</code> .

Value

a list with the fitted model, including the coefficients and their variances for each alpha value.

Examples

```
# Note: example takes approximately 30 seconds to run.

library(survival)
library(dplyr)
library(splines)
# Create followup data with lags
# added variables `..prev_time..`, `..delta_time..` and `..prev_outcome..`
# have special interpretations when computing the influence.
data_with_lags <- SensIAT_example_data |>
  dplyr::group_by(Subject_ID) |>
  dplyr::mutate(
    ..prev_outcome.. = dplyr::lag(Outcome, default = NA_real_, order_by = Time),
    ..prev_time.. = dplyr::lag(Time, default = 0, order_by = Time),
    ..delta_time.. = Time - dplyr::lag(.data$Time, default = NA_real_, order_by = Time)
  )

# Create the observation time intensity model
intensity.model <-
  coxph(Surv(..prev_time.., Time, !is.na(Outcome)) ~ ..prev_outcome.. + strata(Visit),
    data = data_with_lags |> dplyr::filter(.data$Time > 0)
  )

# Create the observed outcome model
outcome.model <-
  fit_SensIAT_single_index_fixed_coef_model(
```

```

      Outcome ~ ns(..prev_outcome.., df = 3) + ..delta_time.. - 1,
      id = Subject_ID,
      data = data_with_lags |> filter(Time > 0)
    )

# Fit the marginal outcome model
mm <- fit_SensIAT_marginal_mean_model(
  data = data_with_lags,
  id = Subject_ID,
  alpha = c(-0.6, -0.3, 0, 0.3, 0.6),
  knots = c(60, 260, 460),
  intensity.model = intensity.model,
  time.vars = c("..delta_time.."),
  outcome.model = outcome.model
)

```

fit_SensIAT_marginal_mean_model_generalized

Fit the marginal mean model for generalize outcomes.

Description

This function supports multiple integration methods for term2 computation, including adaptive and fixed-grid approaches. The implementation includes numerical stability improvements (exp(-) multiplication vs division) and extensive caching optimizations for repeated expected value computations.

Usage

```

fit_SensIAT_marginal_mean_model_generalized(
  data,
  time,
  id,
  alpha,
  knots,
  outcome.model,
  intensity.model,
  impute_data,
  loss = c("lp_mse", "quasi-likelihood"),
  link = c("identity", "log", "logit"),
  spline.degree = 3L,
  ...,
  BBSolve.control = list(maxit = 1000, tol = 1e-06, trace = FALSE),
  term2_method = c("fast", "original", "fixed_grid", "seeded_adaptive", "gauss_legendre"),
  term2_grid_n = 100,
  use_expected_cache = TRUE
)

```

Arguments

data	Data for evaluation of the model. Should match the data used to fit the intensity and outcome models.
time	The time variable in the data. Can be provided as a column name or vector.
id	The subject identifier variable in the data. Lazy evaluation is used, so it can be a symbol or a string.
alpha	Sensitivity parameter, a vector of values.
knots	Location of spline knots. If a SplineBasis object is provided, it is used directly.
outcome.model	The observed effects model.
intensity.model	The assessment time intensity model.
impute_data	A function that takes (t, df) and returns the imputed data at time t. Should handle extrapolation from the last observed time point.
loss	The loss function to use. Options are "lp_mse", "mean_mse", and "quasi-likelihood".
link	The link function to use. Options are "identity", "log", and "logit".
spline.degree	The degree of the spline basis, default is 3 (cubic splines).
...	Additional arguments passed to compute_influence_terms .
BBsolve.control	Control parameters for the BB::sane optimizer, including maxit and tol .
term2_method	Method for computing term2 influence components. Options are: • "fast": Optimized closure-based integrand with adaptive Simpson's (default) • "original": Standard implementation with adaptive Simpson's • "fixed_grid": Pre-computed expected values on fixed grid with composite Simpson's rule • "seeded_adaptive": Adaptive Simpson's seeded with pre-computed grid points • "gauss_legendre": Gauss-Legendre quadrature (requires statmod package)
term2_grid_n	Number of grid points/nodes for fixed_grid , seeded_adaptive , and gauss_legendre methods (default 100)
use_expected_cache	Logical; whether to cache expected values for performance (default TRUE)

Details**Integration Methods for Term2:**

The function offers four integration methods with different performance/accuracy trade-offs:

Adaptive Methods (fast, original):

- Use adaptive Simpson's quadrature with automatic subdivision
- Best accuracy for irregular integrands
- **fast** method uses optimized closure-based integrand construction

Fixed-Grid Method (fixed_grid):

- Pre-computes expected values at fixed grid points (once per alpha)
- Uses composite Simpson's rule for integration
- 2-5x faster when optimizing over beta (multiple iterations)
- Best for smooth integrands with sufficient grid density

Seeded Adaptive Method (seeded_adaptive):

- Combines pre-computation with adaptive refinement
- Starts with pre-computed grid, subdivides where needed
- Good balance of speed and accuracy

Gauss-Legendre Method (gauss_legendre):

- Uses Gauss-Legendre quadrature via `statmod::gauss.quad()`
- Highly accurate for smooth integrands with fewer evaluation points
- Exact for polynomials up to degree $2n-1$ using n points
- Requires the **statmod** package

Outcome Model Compatibility:

Unlike simulation code that assumes specific single-index model formulas, this function supports any outcome model with a `compute_SensIAT_expected_values` method, including:

- Single-index models (`fit_SensIAT_single_index*_model`)
- Generalized linear models (GLM)
- Linear models (LM)
- Negative binomial models

Performance Optimizations:

Term1 Optimizations (alpha-independent):

- Y-scaled observations pre-computed once
- Patient index mappings cached for $O(1)$ lookups
- Identity link: weights pre-computed (don't depend on beta or alpha)
- Single-index models: global PMF constants extracted once

Term2 Optimizations:

- Integration grids pre-computed (alpha-independent)
- Basis evaluations at grid points pre-computed
- Expected values computed once per alpha (for grid methods)
- Per-patient caching of expected values
- Weight functions use numerically stable $\exp(-)$ multiplication
- Weight functions use numerically stable $\exp(-)$ multiplication
- Fast method uses closure-based integration with reduced allocations

Examples

```

library(survival)
library(splines)

data_with_lags <- SensIAT_example_data |>
  dplyr::group_by(Subject_ID) |>
  dplyr::mutate(
    ..prev_outcome.. = dplyr::lag(Outcome, default = NA_real_, order_by = Time),
    ..prev_time.. = dplyr::lag(Time, default = 0, order_by = Time),
    ..delta_time.. = Time - dplyr::lag(.data$Time, default = NA_real_, order_by = Time)
  )

# Create the observation time intensity model
intensity.model <-
  coxph(Surv(..prev_time.., Time, !is.na(Outcome)) ~ ..prev_outcome.. + strata(Visit),
    data = data_with_lags |> dplyr::filter(.data$Time > 0)
  )

# Create the observed outcome model
outcome.model <-
  fit_SensIAT_single_index_fixed_coef_model(
    Outcome ~ ns(..prev_outcome.., df = 3) + ..delta_time.. - 1,
    id = Subject_ID,
    data = data_with_lags |> dplyr::filter(.data$Time > 0)
  )

fit_SensIAT_marginal_mean_model_generalized(
  data = data_with_lags,
  time = data_with_lags$Time,
  id = data_with_lags$Subject_ID,
  alpha = 0,
  knots = c(60, 260, 460),
  outcome.model = outcome.model,
  intensity.model = intensity.model,
  loss = "lp_mse",
  link = "log",
  impute_data = \(t, df){
    data_wl <- df |>
      dplyr::mutate(
        ..prev_time.. = Time,
        ..prev_outcome.. = Outcome,
        ..delta_time.. = 0
      )
    extrapolate_from_last_observation(t, data_wl, "Time", slopes = c("..delta_time.." = 1))
  }
)

```

```
fit_SensIAT_single_index_fixed_coef_model
```

Outcome Modeler for SensIAT Single Index Model.

Description

Outcome Modeler for SensIAT Single Index Model.

Usage

```
fit_SensIAT_single_index_fixed_coef_model(
  formula,
  data,
  kernel = "K2_Biweight",
  method = "nmk",
  id = ..id..,
  initial = NULL,
  ...
)

fit_SensIAT_single_index_fixed_bandwidth_model(
  formula,
  data,
  kernel = "K2_Biweight",
  method = "nmk",
  id = ..id..,
  initial = NULL,
  ...
)
```

Arguments

formula	The outcome model formula
data	The data to fit the outcome model to. Should only include follow-up data, i.e. time > 0.
kernel	The kernel to use for the outcome model.
method	The optimization method to use for the outcome model, either "optim", "nlminb", or "nmk".
id	The patient identifier variable for the data.
initial	Either a vector of initial values or a function to estimate initial values. If NULL (default), the initial values are estimated using the <code>MAVE::mave.compute</code> function.
...	Currently ignored, included for future compatibility.

Value

Object of class `SensIAT::Single-index-outcome-model` which contains the outcome model portion.

Functions

- `fit_SensIAT_single_index_fixed_bandwidth_model()`: for fitting with a fixed bandwidth

Examples

```
# A basic example using fixed intensity bandwidth.
object <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_bandwidth_model,
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    knots = c(60, 260, 460),
    End = 830,
    intensity.args = list(bandwidth = 30)
  )

# A basic example using variable bandwidth but with fixed first coefficient.
object.bw <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    knots = c(60, 260, 460),
    End = 830,
    intensity.args = list(bandwidth = 30)
  )
```

```
fit_SensIAT_single_index_norm1coef_model
```

Single Index Model using MAVE and Optimizing Bandwidth.

Description

Single index model estimation using minimum average variance estimation (MAVE). A direction is estimated using MAVE, and then the bandwidth is selected by minimization of the cross-validated pseudo-integrated squared error. Optionally, the initial coefficients of the outcome model can be re-estimated by optimization on a spherical manifold. This option requires the **ManifoldOptim** package (see [ManifoldOptim::manifold.optim\(\)](#)).

Usage

```
fit_SensIAT_single_index_norm1coef_model(
  formula,
  data,
  kernel = "K2_Biweight",
  mave.method = "meanMAVE",
  id = ..id..,
  bw.selection = c("ise", "mse"),
  bw.method = c("optim", "grid", "optimize"),
  bw.range = c(0.01, 1.5),
  reestimate.coef = 0,
  ...
)
```

Arguments

<code>formula</code>	The outcome model formula
<code>data</code>	The data to fit the outcome model to. Should only include follow-up data, i.e. <code>time > 0</code> .
<code>kernel</code>	The kernel to use for the outcome model.
<code>mave.method</code>	The method to use for the MAVE estimation.
<code>id</code>	The patient identifier variable for the data.
<code>bw.selection</code>	The criteria for bandwidth selection, either <code>'ise'</code> for Integrated Squared Error or <code>'mse'</code> for Mean Squared Error.
<code>bw.method</code>	The method for bandwidth selection, either <code>'optim'</code> for using optimization or <code>'grid'</code> for grid search.
<code>bw.range</code>	A numeric vector of length 2 indicating the range of bandwidths to consider for selection as a multiple of the standard deviation of the single index predictor.
<code>reestimate.coef</code>	number of iterations to go through.
<code>...</code>	Additional arguments to be passed to optim .

Value

Object of class `SensIAT::Single-index-outcome-model` which contains the outcome model portion.

`get_bootstrap_coeffs` *Get coefficients for bootstrap simulation.*

Description

By default this returns original fitted coefficients. If sampling is requested, coefficients are sampled from an asymptotic multivariate normal distribution only when `vcov()` is available.

Usage

```
get_bootstrap_coeffs(model, sample_coefficients = FALSE, model_label = "model")
```

Arguments

<code>model</code>	A fitted model with <code>coef()</code> and <code>vcov()</code> methods.
<code>sample_coefficients</code>	Logical; if <code>TRUE</code> , sample coefficients from an asymptotic multivariate normal distribution when <code>vcov()</code> is available. If <code>FALSE</code> (default), use original fitted coefficients.
<code>model_label</code>	Character; label for the model used in warning messages.

<code>jackknife</code>	<i>Perform Jackknife Resampling on an Object</i>
------------------------	--

Description

Perform Jackknife Resampling on an Object

Usage

```
jackknife(object, ...)

## S3 method for class 'SensIAT_within_group_model'
jackknife(object, time, progress = interactive(), ...)

## S3 method for class 'SensIAT_fullldata_model'
jackknife(object, time, ...)
```

Arguments

<code>object</code>	An object to cross validate on.
<code>...</code>	Additional arguments passed to the method.
<code>time</code>	Time points for which to estimate the response.
<code>progress</code>	Logical. Show a progress bar while fitting leave-one-out replications. Defaults to <code>TRUE</code> when called interactively.

Value

A data frame of the jackknife resampling results. For `SensIAT` objects, a `tibble` with columns `alpha`, `time`, `jackknife_mean`, and `jackknife_var`, where `jackknife_mean` is the mean of the jackknife estimates and `jackknife_var` is the estimated variances of the response at the given time points for the specified `alpha` values.

Methods (by class)

- `jackknife(SensIAT_within_group_model)`: Perform jackknife resampling on a `SensIAT_within_group_model` object.
- `jackknife(SensIAT_fulldata_model)`: Perform jackknife resampling on a `SensIAT_fulldata_model` object.

Examples

```
## Not run:
object <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6),
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    intensity.args = list(bandwidth = 30),
    knots = c(60, 260, 460),
    End = 830
  )
jackknife.estimates <- jackknife(object, time = c(90, 180, 270, 360, 450))

## End(Not run)
```

```
make_parametric_intensity_simulator
```

Build a parametric intensity simulator using sampled coefficients

Description

Supports `coxph` objects: sampled coefficients are used to compute linear predictor while baseline hazard is taken from `survival::basehaz()`.

Usage

```
make_parametric_intensity_simulator(
  intensity_model,
  sampled_coef,
  covariate_mapping = NULL
)
```

Arguments

intensity_model A fitted model (e.g., `coxph`) or a `function(t, prev_outcome, visit_num)`.

sampled_coef Named numeric vector of sampled coefficients (from `get_bootstrap_coeffs()`).

covariate_mapping

Optional named character vector mapping expected covariate names to model variable names.

Value

A function of signature `function(t, prev_outcome, visit_num)` returning non-negative numeric intensity.

[Experimental]

make_parametric_single_index_simulator

Build a parametric outcome simulator from a fitted single-index outcome model using sampled coefficients for the single-index projection.

Description

Build a parametric outcome simulator from a fitted single-index outcome model using sampled coefficients for the single-index projection.

Usage

```
make_parametric_single_index_simulator(
  outcome_model,
  sampled_coef,
  covariate_mapping = NULL
)
```

Arguments

`outcome_model` A fitted `SensIAT::Single-index-outcome-model` object.

`sampled_coef` Named numeric vector of sampled coefficients (from `get_bootstrap_coeffs()`).

`covariate_mapping`

Optional named character vector mapping expected covariate names to model variable names. **[Experimental]**

```
make_single_index_simulator
```

Create a simulator function from a fitted Single-index outcome model

Description

This factory builds a closure that can be used to simulate outcome values from a fitted `SensIAT::Single-index-outcome-model`. The returned function samples from the estimated conditional distribution of the outcome given covariates using the Nadaraya-Watson estimator implemented in `pcoriaccel_NW`.

Usage

```
make_single_index_simulator(outcome_model, covariate_mapping = NULL)
```

Arguments

`outcome_model` A fitted object of class `SensIAT::Single-index-outcome-model`.

`covariate_mapping`

Optional named character vector mapping expected covariate names (e.g. `prev_outcome`, `time`, `delta_time`) to the names used in the original model formula. If `NULL`, the factory will attempt to use `prev_outcome`, `time`, and `delta_time` directly.

Value

A function with signature `function(prev_outcome, time, delta_time, newdata = NULL)` which returns a sampled outcome value consistent with the fitted model.

```
parametric_bootstrap Parametric bootstrap orchestration [Experimental]
```

Description

Parametric bootstrap orchestration [Experimental]

Usage

```
parametric_bootstrap(
  nboot = 100,
  intensity_model = NULL,
  outcome_model = NULL,
  simulate_args = list(),
  seed = NULL,
  progress = interactive(),
```

```

    sample_coefficients = FALSE,
    verbosity = c("none", "basic", "detailed"),
    verbose = NULL
  )

```

Arguments

<code>nboot</code>	Number of bootstrap replicates
<code>intensity_model</code>	Fitted intensity model (survival::coxph) or function
<code>outcome_model</code>	Fitted outcome model (single-index) or NULL
<code>simulate_args</code>	List of arguments to pass to <code>simulate_SensIAT_data()</code> (e.g., <code>n_subjects</code> , <code>End</code> , <code>intensity_bound</code>)
<code>seed</code>	Optional seed for reproducibility
<code>progress</code>	Logical; show progress bar when available.
<code>sample_coefficients</code>	Logical; if TRUE, sample coefficients from an asymptotic multivariate normal distribution when <code>vcov()</code> is available. If FALSE (default), use original fitted coefficients.
<code>verbosity</code>	Logging verbosity for bootstrap internals: one of "none", "basic", or "detailed".
<code>verbose</code>	Deprecated shortcut; if TRUE, equivalent to <code>verbosity = "detailed"</code> .

Value

A list of simulated datasets (length `nboot`)

`parametric_bootstrap_within_group`

Parametric bootstrap for a within-group SensIAT model **[Experimental]**

Description

Parametric bootstrap for a within-group SensIAT model **[Experimental]**

Usage

```

parametric_bootstrap_within_group(
  within_group_model,
  nboot = 100,
  simulate_args = list(),
  seed = NULL,
  progress = interactive(),
  sample_coefficients = FALSE,

```

```

    refit = TRUE,
    return = c("coefficients", "models", "data"),
    prune_models = FALSE,
    gc_every = 10L,
    verbosity = c("none", "basic", "detailed"),
    verbose = NULL
  )

```

Arguments

<code>within_group_model</code>	A fitted <code>SensIAT_within_group_model</code> object.
<code>nboot</code>	Number of bootstrap replicates.
<code>simulate_args</code>	List of arguments to pass to <code>simulate_SensIAT_data()</code> . If not specified, <code>End</code> , <code>n_subjects</code> , <code>initial_outcome_mean</code> , and <code>initial_outcome_sd</code> are inferred from the fitted model.
<code>seed</code>	Optional seed for reproducibility.
<code>progress</code>	Logical; show progress bar when available.
<code>sample_coefficients</code>	Logical; if <code>TRUE</code> , sample coefficients from an asymptotic multivariate normal distribution when <code>vcov()</code> is available. If <code>FALSE</code> (default), use original fitted coefficients.
<code>refit</code>	Logical; if <code>TRUE</code> (default), fit a <code>SensIAT_within_group_model</code> on each simulated replicate using the original model's settings.
<code>return</code>	One of <code>"coefficients"</code> (default), <code>"models"</code> , or <code>"data"</code> . <code>"coefficients"</code> is memory-efficient and stores only replicated marginal mean coefficients.
<code>prune_models</code>	Logical; when <code>return = "models"</code> , prune each replicated model before returning.
<code>gc_every</code>	Integer. Run <code>gc(FALSE)</code> every <code>gc_every</code> replications. Use <code>NULL</code> to disable explicit garbage collection.
<code>verbosity</code>	Logging verbosity for bootstrap internals: one of <code>"none"</code> , <code>"basic"</code> , or <code>"detailed"</code> .
<code>verbose</code>	Deprecated shortcut; if <code>TRUE</code> , equivalent to <code>verbosity = "detailed"</code> .

Value

If `return = "coefficients"`, a `SensIAT_withingroup_bootstrap_results` object. Otherwise returns a list of replicated fitted models (`"models"`) or simulated datasets (`"data"`).

Examples

```

## Not run:
data("SensIAT_example_data", package = "SensIAT")

# Fit a single-index outcome model on a small subset of the example data.
small_data <- dplyr::filter(
  SensIAT_example_data,

```

```

  Subject_ID %in% head(unique(SensIAT_example_data$Subject_ID), 8)
)

model <- fit_SensIAT_within_group_model(
  group.data = small_data,
  outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
  alpha = 0,
  id = Subject_ID,
  outcome = Outcome,
  time = Time,
  End = 830,
  knots = c(60, 260, 460)
)

# This example may take a long time because it fits a single-index outcome model
# and generates bootstrap replicates.
res <- parametric_bootstrap_within_group(
  nboot = 2,
  within_group_model = model,
  simulate_args = list(
    n_subjects = 3,
    End = 5,
    max_visits = 5
  ),
  seed = 123,
  refit = TRUE
)

print(res[[1]])

## End(Not run)

```

```
predict.SensIAT_fulldata_model
```

Give the Marginal Mean Estimate and its Estimated Asymptotic Variance

Description

Give the marginal mean model estimate

Usage

```

## S3 method for class 'SensIAT_fulldata_model'
predict(object, time, ...)

## S3 method for class 'SensIAT_within_group_model'
predict(object, time, include.var = TRUE, ..., base = object$base)

```

Arguments

object	SensIAT_within_group_model object
time	Time points of interest
...	Currently ignored.
include.var	Logical. If TRUE, the variance of the outcome is also returned
base	A SplineBasis object used to evaluate the basis functions.

Value

If include.var is TRUE and the fitted model uses the identity link, a **tibble** with columns **time**, **mean**, and **var** is returned. For non-identity links, variance estimation is not supported and only **time** and **mean** are returned. If include.var is FALSE, only the mean vector is returned.

Functions

- **predict(SensIAT_fulldata_model)**: For each combination of **time** and **alpha** estimate the mean response and variance for each group as well as estimate the mean treatment effect and variance.

Examples

```
model <-
  fit_SensIAT_within_group_model(
    group.data = SensIAT_example_data,
    outcome_modeler = fit_SensIAT_single_index_fixed_coef_model,
    alpha = c(-0.6, -0.3, 0, 0.3, 0.6),
    id = Subject_ID,
    outcome = Outcome,
    time = Time,
    End = 830,
    knots = c(60, 260, 460),
  )
predict(model, time = c(90, 180))
```

```
predict.SensIAT_within_group_bootstrap_results
```

Predict Marginal Mean from Bootstrap Coefficient Replicates

Description

Predict Marginal Mean from Bootstrap Coefficient Replicates

Usage

```
## S3 method for class 'SensIAT_withingroup_bootstrap_results'
predict(object, time, include.var = TRUE, level = 0.95, ...)
```

Arguments

<code>object</code>	A <code>SensIAT_withingroup_bootstrap_results</code> object.
<code>time</code>	Time points of interest.
<code>include.var</code>	Logical. If <code>TRUE</code> and link is identity, include the original-model asymptotic variance column <code>var</code> .
<code>level</code>	Confidence level used to produce lower and upper bounds.
<code>...</code>	Currently ignored.

Value

A tibble with bootstrap summaries by `alpha` and `time`.

<code>prepare_SensIAT_data</code>	<i>Prepare Data for Sensitivity Analysis with Irregular Assessment Times</i>
-----------------------------------	--

Description

This function prepares the data for SensIAT analysis by transforming it into a format suitable for the SensIAT models.

Usage

```
prepare_SensIAT_data(
  data,
  id.var,
  time.var,
  outcome.var,
  End,
  add.terminal.observations = TRUE
)
```

Arguments

<code>data</code>	A data frame containing the data to be prepared.
<code>id.var</code>	The variable in <code>data</code> that identifies the subject.
<code>time.var</code>	The variable in <code>data</code> that identifies the time of the observation.
<code>outcome.var</code>	The variable in <code>data</code> that contains the outcome of interest.
<code>End</code>	The end time for the analysis. Observations with time greater than <code>End</code> will be filtered out.
<code>add.terminal.observations</code>	Logical indicating whether to add terminal observations to the data (<code>TRUE</code>), or terminal observations have already been added (<code>FALSE</code>).

Value

A data frame with the following transformations:

- Data filtered to time less than or equal to `End`.
- Observations are arranged by `id.var` and `time.var`.
- Terminal observations added if `add.terminal.observations` is `TRUE`, with `..time..` set to `End` and `..outcome..` set to `NA`, if the subject has less observations than the maximum number of observations.
- New variables created:
 - `..id..` aliases `id.var`,
 - `..time..` aliases `time.var`,
 - `..outcome..` aliases `outcome.var`,
 - `..visit_number..` is the visit number within each subject derived from `time.var`,
 - `..prev_outcome..`, i.e. lag-outcome, the outcome from the previous visit,
 - `..prev_time..`, i.e. lag-time, the time from the previous visit,
 - `..delta_time..`, the difference in time between the current and previous visit.

Examples

```
prepare_SensIAT_data(SensIAT_example_data, Subject_ID, Time, Outcome, 830)

exdata <- tibble::tibble(
  ID = rep(1:2, c(3, 5)),
  Time = c(
    0, 30, 60,
    0, 30, 60, 90, 120
  ),
  Outcome = floor(runif(8, 1, 100))
)

prepare_SensIAT_data(exdata, ID, Time, Outcome, 120)
```

`sample_parametric_coeffs`

Sample parametric coefficients from a fitted model using asymptotic multivariate normal distribution.

Description

Sample parametric coefficients from a fitted model using asymptotic multivariate normal distribution.

Usage

```
sample_parametric_coeffs(model)
```

Arguments

model A fitted model with `coef()` and `vcov()` methods

Value

Named numeric vector of sampled coefficients

SensIAT_example_data *SensIAT Example Data*

Description

A simulated dataset for use in the SensIAT tutorial, testing and documentation.

Usage

`SensIAT_example_data`

`SensIAT_example_fulldata`

Format

`SensIAT_example_data` is a data frame with 779 rows and 4 variables consisting of 200 simulated patients. Each row in the data represents a visit for the patient. The columns are:

Subject_ID A unique identifier for each patient.

Visit The ordinal number of the visit for the patient. Baseline observation is 0.

Time The time of the visit in days, since baseline.

Outcome The outcome of interest.

`SensIAT_example_fulldata` is a data frame with 1614 rows and 5 variables consisting of 400 simulated patients, 200 for each treatment arm. Each row in the data represents a visit for the patient. The columns are:

Subject_ID A unique identifier for each patient.

Visit The ordinal number of the visit for the patient. Baseline observation is 0.

Time The time of the visit in days, since baseline.

Outcome The outcome of interest.

Treatment_group Treatment or control group.

Functions

- `SensIAT_example_fulldata`: A simulated dataset with both treatment and control groups.

simulate_SensIAT_data

Simulate SensIAT Data

Description

Generates simulated longitudinal data following the SensIAT model structure, alternating between observation time generation (Cox proportional hazards model) and outcome generation.

Usage

```
simulate_SensIAT_data(
  n_subjects,
  End,
  intensity_coef = -0.05,
  outcome_coef = list(prev_outcome = c(0.7, -0.1, 0.05), time = -0.001, delta_time =
    -0.002, intercept = 2),
  baseline_hazard = 0.005,
  outcome_sd = 1.5,
  baseline_outcome_fn = NULL,
  initial_outcome_mean = 5,
  initial_outcome_sd = 2,
  max_visits = 50,
  seed = NULL,
  link = "identity",
  intensity_fn = NULL,
  intensity_bound = NULL,
  outcome_model = NULL,
  outcome_simulator = NULL
)
```

Arguments

<code>n_subjects</code>	Number of subjects to simulate.
<code>End</code>	Maximum follow-up time.
<code>intensity_coef</code>	Coefficient for the effect of previous outcome on observation intensity. Can be a scalar or vector (one per visit number stratum).
<code>outcome_coef</code>	Named list of coefficients for outcome model including: <code>prev_outcome</code> - coefficients for natural spline of previous outcome (length 3) <code>time</code> - coefficient for time since baseline <code>delta_time</code> - coefficient for time since last observation <code>intercept</code> - intercept term

baseline_hazard	Baseline hazard function. Either a function of time and visit number, or a numeric value for constant baseline hazard.
outcome_sd	Standard deviation of the outcome residuals.
baseline_outcome_fn	Optional function to generate baseline outcome value for each subject.
initial_outcome_mean	Mean of the initial (baseline) outcome.
initial_outcome_sd	Standard deviation of the initial outcome.
max_visits	Maximum number of visits per subject (to prevent infinite loops).
seed	Random seed for reproducibility.
link	Link function for outcome model. One of "identity", "log", or "logit". Determines the scale on which the outcome model operates.
intensity_fn	Optional function to compute intensity (hazard) of observation. If provided, should take arguments (time , prev_outcome , visit_num) and return a scalar intensity value. If NULL (default), intensity is computed from intensity_coef and baseline_hazard .
intensity_bound	Upper bound on intensity for rejection sampling. Required if intensity_fn is provided. Represents the supremum of the intensity function on the interval of interest.
outcome_model	Optional fitted single-index outcome model. If provided, outcomes for follow-up visits are generated from the fitted model via make_single_index_simulator() .
outcome_simulator	Optional simulator function for follow-up outcomes. When provided, it overrides the internal outcome generation function. This function should accept prev_outcome , time , delta_time , and optionally newdata .

Value

A tibble with columns:

- **Subject_ID** - Subject identifier
- **Time** - Observation time
- **Outcome** - Observed outcome value
- Additional columns may be added for internal use

Examples

```
# Default usage (uses exponential gaps derived from intensity_coef and baseline_hazard)
sim_data <- simulate_SensIAT_data(
  n_subjects = 100,
  End = 830,
  intensity_coef = -0.05,
  outcome_coef = list()
```

```

        prev_outcome = c(0.7, -0.1, 0.05),
        time = -0.001,
        delta_time = -0.002,
        intercept = 2
    ),
    baseline_hazard = 0.005,
    outcome_sd = 1.5
)

# Example with custom intensity function and thinning (Exp(lambda_star))
intensity_fn <- function(t, prev_outcome, visit_num) {
  lambda0 <- 0.005
  gamma <- -0.05
  lambda0 * exp(gamma * prev_outcome)
}
sim_data2 <- simulate_SensIAT_data(
  n_subjects = 50,
  End = 200,
  seed = 123,
  intensity_fn = intensity_fn,
  intensity_bound = 0.05,
  max_visits = 20
)

# Example using a fitted single-index outcome model to generate outcomes
# via the fitted conditional distribution.
#
# Note: this example uses a fitted model object and is intended for
# parametric bootstrap-style simulation.
#
#
# outcome_model <- fit_SensIAT_single_index_fixed_coef_model(
#   Outcome ~ prev_outcome + time + delta_time,
#   data = training_data,
#   id = Subject_ID
# )
# sim_data3 <- simulate_SensIAT_data(
#   n_subjects = 50,
#   End = 200,
#   seed = 123,
#   outcome_model = outcome_model,
#   intensity_fn = intensity_fn,
#   intensity_bound = 0.05,
#   max_visits = 20
# )

```

simulate_SensIAT_two_groups

Simulate Treatment and Control Groups

Description

Generate simulated data for both treatment and control groups with potentially different parameters.

Usage

```
simulate_SensIAT_two_groups(
  n_subjects,
  End,
  intensity_coef = -0.05,
  outcome_coef = list(prev_outcome = c(0.7, -0.1, 0.05), time = -0.001, delta_time =
    -0.002, intercept = 2),
  baseline_hazard = 0.005,
  outcome_sd = 1.5,
  initial_outcome_mean = 5,
  initial_outcome_sd = 2,
  max_visits = 50,
  treatment_effect = 0,
  treatment_intensity_effect = 1,
  seed = NULL,
  link = "identity",
  intensity_fn = NULL,
  intensity_bound = NULL,
  outcome_model = NULL,
  outcome_simulator = NULL
)
```

Arguments

n_subjects	Number of subjects to simulate.
End	Maximum follow-up time.
intensity_coef	Coefficient for the effect of previous outcome on observation intensity. Can be a scalar or vector (one per visit number stratum).
outcome_coef	Named list of coefficients for outcome model including: • prev_outcome - coefficients for natural spline of previous outcome (length 3) • time - coefficient for time since baseline • delta_time - coefficient for time since last observation • intercept - intercept term
baseline_hazard	Baseline hazard function. Either a function of time and visit number, or a numeric value for constant baseline hazard.
outcome_sd	Standard deviation of the outcome residuals.
initial_outcome_mean	Mean of the initial (baseline) outcome.

<code>initial_outcome_sd</code>	Standard deviation of the initial outcome.
<code>max_visits</code>	Maximum number of visits per subject (to prevent infinite loops).
<code>treatment_effect</code>	Additive treatment effect on outcomes (added to intercept on link scale).
<code>treatment_intensity_effect</code>	Multiplicative effect on observation intensity (values < 1 mean fewer observations in treatment group).
<code>seed</code>	Random seed for reproducibility.
<code>link</code>	Link function for outcome model. One of "identity", "log", or "logit".
<code>intensity_fn</code>	Optional function to compute intensity (hazard) of observation. If provided, should take arguments (<code>time</code> , <code>prev_outcome</code> , <code>visit_num</code>) and return a scalar intensity value. If NULL (default), intensity is computed from <code>intensity_coef</code> and <code>baseline_hazard</code> .
<code>intensity_bound</code>	Upper bound on intensity for rejection sampling. Required if <code>intensity_fn</code> is provided. Represents the supremum of the intensity function on the interval of interest.
<code>outcome_model</code>	Optional fitted single-index outcome model. If provided, outcomes for follow-up visits are generated from the fitted model via <code>make_single_index_simulator()</code> .
<code>outcome_simulator</code>	Optional simulator function for follow-up outcomes. When provided, it overrides the internal outcome generation function. This function should accept <code>prev_outcome</code> , <code>time</code> , <code>delta_time</code> , and optionally <code>newdata</code> .

Value

A tibble with an additional `Treatment` column indicating group assignment.

Examples

```
# Default treatment/control simulation (uses exponential gaps derived from coefficients)
sim_data <- simulate_SensIAT_two_groups(
  n_subjects = 100,
  End = 830,
  treatment_effect = 1.5,
  treatment_intensity_effect = 0.9
)

# Example using custom intensity with thinning
intensity_fn <- function(t, prev_outcome, visit_num) {
  lambda0 <- 0.005
  gamma <- -0.05
  lambda0 * exp(gamma * prev_outcome)
}
sim_data2 <- simulate_SensIAT_two_groups(
  n_subjects = 100,
  End = 200,
```

```
seed = 123,  
intensity_fn = intensity_fn,  
intensity_bound = 0.05,  
treatment_effect = 1.0,  
treatment_intensity_effect = 0.9  
)
```

Index

* datasets
 SensIAT_example_data, 36
 autoplot.SensIAT_fulldata_jackknife_results, 3
 autoplot.SensIAT_fulldata_model, 4
 autoplot.SensIAT_within_group_model, 5
 autoplot.SensIAT_withingroup_bootstrap_results, 6
 autoplot.SensIAT_withingroup_jackknife_results, 6
 benchmark_term2_methods, 7
 compute_influence_terms, 9
 compute_SensIAT_expected_values, 11
 fit_marginal_model, 13
 fit_SensIAT_fulldata_model, 14
 fit_SensIAT_marginal_mean_model, 17
 fit_SensIAT_marginal_mean_model_generalized, 19
 fit_SensIAT_single_index_fixed_bandwidth_model, 23
 (*fit_SensIAT_single_index_fixed_coef_model*),
 fit_SensIAT_single_index_fixed_coef_model, 22
 fit_SensIAT_single_index_fixed_coef_model(), 8
 fit_SensIAT_single_index_norm1coef_model, 24
 fit_SensIAT_within_group_model
 (*fit_SensIAT_fulldata_model*),
 get_bootstrap_coeffs, 25
 jackknife, 26
 make_parametric_intensity_simulator, 27
 make_parametric_single_index_simulator, 28
 make_single_index_simulator, 29
 make_single_index_simulator(), 38, 41
 ManifoldOptim::manifold.optim(), 24
 optim, 25
 parametric_bootstrap, 29
 parametric_bootstrap_within_group, 30
 predict.SensIAT_fulldata_model, 32
 predict.SensIAT_within_group_model
 (*predict.SensIAT_fulldata_model*),
 predict.SensIAT_withingroup_bootstrap_results, 33
 prepare_SensIAT_data, 34
 sample_parametric_coeffs, 35
 SensIAT_example_data, 36
 SensIAT_example_fulldata
 (*SensIAT_example_data*), 36
 simulate_SensIAT_data, 37
 simulate_SensIAT_two_groups, 39
 SplineBasis, 16
 statmod::gauss.quad(), 21
 survival::coxph, 30
 update.formula, 15, 16