Package 'LAGOtrials'

November 11, 2026

Type Package

Title Learn-as-You-Go Adaptive Trial Optimization

Version 1.2.0

Description Optimizes interventions for adaptive ``Learn-As-you-GO'' (LAGO) trials, in which the intervention package is updated between stages using accumulating data. Given data from completed stages, lago_optimization() fits the outcome model (via 'stats' generalized linear models or, for clustered designs, cluster-robust variance) and computes the cost-optimal recommended intervention for the next stage that is expected to reach a target mean outcome and/or a desired statistical power. get_confidence_set() constructs confidence sets for the optimal intervention, and visualize_cost() and lago_report() summarize the cost surface and results. Methods are described in Nevo, Lok and Spiegelman (2021) <doi:10.1214/20-aos1978> and Bing, Spiegelman, Nevo and Lok (2025) <doi:10.1093/biomtc/ujaf061>.

License GPL-3

Copyright This package bundles the D3 v7 library (inst/js/d3.v7.min.js), which is ISC-licensed and Copyright 2010-2023 Mike Bostock. See inst/COPYRIGHTS and inst/js/d3-LICENSE. All other code is Copyright the LAGO authors under GPL-3.

URL https://correspondmerchant.github.io/LAGO-R-Package/, https://github.com/correspondMerchant/LAGO-R-Package

BugReports https://github.com/correspondMerchant/LAGO-R-Package/issues

Encoding UTF-8

LazyData true

Imports rje, NlcOptim, cli, bslib (>= 0.7.0), shiny (>= 1.8.1), htmltools, ggplot2, shinyjs, Rcpp, later

LinkingTo Rcpp

Suggests testthat (>= 3.0.0), tibble, knitr, rmarkdown, jsonlite, spelling

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 2.10)

Config/roxygen2/version 8.0.0

Language en-US

NeedsCompilation yes

Author Ante Bing [aut, cre],
Minh Bui [aut],
Jingyu Cui [aut],
Mike Bostock [cph] (Author of the bundled D3 v7 library,
inst/js/d3.v7.min.js)

Maintainer Ante Bing <abing@bu.edu>

Repository CRAN

Date/Publication 2026-11-11 07:30:18 UTC

Contents

BB_data 2
BB_proportions 4
lago_app 4
lago_budget 6
lago_optimization 8
lago_report 16
lago_sensitivity 18
main_pulesa_data 21
new_pulesa_data 22
plot.lago 23
plot.lago_budget 24
plot.lago_sensitivity 25
print.lago 26
print.lago_budget 27
print.lago_sensitivity 28
summary.lago 29
visualize_cost 30

Index **32**

`BB_data` *BetterBirth Data (for binary outcome analysis)*

Description

This is the cleaned version of the BetterBirth dataset used in Nevo et al.

Usage

```
BB_data
```

Format

A data frame with 6124 rows and 21 variables:

coaching_updt Number of coaching visits accrued (at time of birth)

data_flag Indicator of original dataset

datafeedback Data feedback loop

launch_duration Duration of checklist launch

leadership_ITT Leadership engagement, ITT

leadership_updt Intensity of leadership engagement (at time of birth)

pre_post Indicator for visits after initial coaching visit

pilot_phase Phase of the pilot studies

site_name Site name

always_checklist Checklist was used in all phases

sometimes_checklist Checklist was only in some phases

birth_volume Estimated monthly birth volume

birth_volume_100 Estimated monthly birth volume (divided by 100)

distance Estimated distance to nearby hospital (km)

distance_10 Estimated distance to nearby hospital (divided by 10)

staff_nurse Number of ANM, staff nurses, and lady health visitors

pp2_hand_hygiene fada_op2_handwashing

pp3_oxytocin_mother fada_op3_oxytocin_administered

center_type Phase of the pilot studies

coach5 Number of coaching vissts accrued divided by 5

coach3 Number of coaching vissts accrued divided by 3

Source

doi:10.1186/s130120150309y

doi:10.1056/NEJMoa1701075

Examples

```
data(BB_data)
head(BB_data)
```

BB_proportions *BetterBirth Data (for continuous outcome analysis, using EBP proportions as a continuous outcome).*

Description

This is the cleaned version of the BetterBirth dataset used in Bing et al.

Usage

```
BB_proportions
```

Format

A data frame with 7359 rows and 4 variables:

coaching_updt Number of coaching visits accrued (at time of birth)

launch_duration Duration of checklist launch

birth_volume_100 Estimated monthly birth volume (divided by 100)

EBP_proportions Essential birth practices performed out of all possible birth practices measured. We consider this as a continuous outcome.

Source

https://arxiv.org/pdf/2307.06552

Examples

```
data(BB_proportions)
head(BB_proportions)
```

lago_app *Run a LAGO optimization in a point and click app*

Description

Opens a Shiny app in your web browser that runs the whole LAGO optimization without writing R code. You load a dataset, choose the outcome and the goal, choose the intervention components with their bounds and costs, and say how the model accounts for centers and time. The app then finds the recommended intervention with [lago_optimization()] and shows the result, the confidence set and the R code that reproduces the run.

Usage

```
lago_app(data = NULL, launch.browser = interactive(), port = NULL)
```

Arguments

`data`	A data.frame to start with, or ‘NULL‘ to start with the bundled ‘BB_data‘ example.
`launch.browser`	Whether to open the app in your web browser. By default it opens in an interactive session.
`port`	The port to listen on, or ‘NULL‘ for a free one.

Details

The app runs on your own computer and is reachable only from it, at ‘127.0.0.1‘. Nothing is sent over the network.

The app has these steps.

* **Data:** the ‘data‘ you pass, the bundled ‘BB_data‘, R’s ‘mtcars‘, or a CSV file you upload. Each example opens on a setup that runs as is. * **Outcome and goal:** ‘outcome_name‘, ‘outcome_type‘, ‘outcome_goal‘, ‘outcome_goal_intention‘, and for a binary outcome the power goal: ‘power_goal‘, ‘num_centers_in_next_stage‘, ‘patients_per_center_in_next_stage‘, ‘power_goal_approach‘, ‘icc‘ and ‘power_goal_cluster_id‘. The app builds the ‘group‘ column from the treatment arm column you choose. * **Intervention and costs:** ‘intervention_components‘, their bounds, and the costs as ‘unit_costs‘ with ‘default_cost_fxn_type‘ and ‘unit_cost_doses‘, or as ‘cost_list_of_vectors‘. Interaction terms set ‘include_interaction_terms‘ and ‘main_components‘, and the app builds each product column. * **Centers, time and covariates:** ‘center_characteristics‘ with their values, or fixed center effects with ‘include_center_effects‘ and ‘center_effects_optimization_values‘, fixed time effects with ‘include_time_effects‘ and ‘time_effect_optimization_value‘, and ‘additional_covariates‘. The app builds the ‘center‘ and ‘period‘ columns from the columns you choose.

A step that needs attention says what to change, and the run button stays off until every step is ready. Warnings from the package are listed with the result.

Value

Invisibly, when you click **Done** or close the browser tab, a list with the last ‘result‘ of [lago_optimization()] (or ‘NULL‘ if nothing was run), its R ‘code‘ as a character vector, and the ‘arguments‘ it was called with. The same list is stored in the ‘lago_app_result‘ option.

See Also

[lago_optimization()] for the function the app runs, and [visualize_cost()] for the cost designer.

Examples

```
if (interactive()) {
  out <- lago_app()
  out$result
  cat(out$code, sep = "\n")
}
```

`lago_budget` *Budget-constrained LAGO optimization*

Description

Answers the reverse of the usual LAGO question. Instead of "what is the least costly intervention that reaches an outcome goal?" ([lago_optimization()]), ‘lago_budget()‘ asks "given a fixed budget, what is the best outcome I can reach, and with which intervention?".

The best-outcome-within-budget intervention always lies on the same least-cost frontier that [lago_optimization()] traces out (for any target outcome, the cheapest way to reach it is what the optimizer already finds), so ‘lago_budget()‘ sweeps the outcome goal across the outcome's range with [lago_sensitivity()], reads the recommended cost and estimated outcome at each goal, and returns the reachable goal whose recommended cost is highest without exceeding the budget (for ‘outcome_goal_intention = "maximize"‘; the lowest reachable outcome within budget for ‘"minimize"‘). It does not touch the optimizer internals, so it inherits the same model, cost functions and bounds.

Usage

```
lago_budget(object = NULL, ..., budget, n_grid = 25L, quiet = TRUE)
```

Arguments

`object`	An optional ‘lago‘ result from [lago_optimization()]. When supplied, the baseline optimization arguments are read from the call it carries, so the whole call need not be retyped, and anything in ‘...‘ overrides those stored values. When ‘NULL‘ (the default), the baseline arguments come from ‘...‘. Passing a non-‘lago‘ object, or a ‘lago‘ result from a version that did not record its call arguments, is an error.
`...`	The baseline [lago_optimization()] arguments (the user's own optimization call), forwarded unchanged to every run. ‘outcome_goal‘ is swept and so need not be supplied; ‘include_confidence_set‘ and ‘quiet‘ are overridden (the confidence set is never computed during the search).
`budget`	A single positive, finite numeric value. The maximum total cost allowed, in the same units as the supplied ‘cost_list_of_vectors‘ (or ‘unit_costs‘).
`n_grid`	A single integer (>= 2) giving how many outcome goals to try across the outcome's range. Larger values give a finer, more precise answer at the cost of more optimization runs. A second refinement pass tries ‘n_grid‘ more goals strictly inside the gap at the affordability boundary, so the effective resolution is finer than ‘n_grid‘ alone. Default 25.
`quiet`	A boolean forwarded to [lago_optimization()] via [lago_sensitivity()]. Defaults to ‘TRUE‘ so the search is not noisy.

Details

The candidate outcome goals span the outcome's range: ‘(0, 1)‘ for a binary outcome, or the observed range of the outcome column (padded by half a span on each side) for a continuous one. Each goal is one [lago_optimization()] run with the confidence set off. A goal beyond the reachable range does not error: the optimizer's shrinking method returns a recommendation whose estimated outcome falls short of the goal, so such a goal is not counted as genuinely reached and does not become a candidate (only goals the recommendation actually delivers are considered). Among the reached goals whose recommended cost is at most ‘budget‘, the one with the best estimated outcome is chosen (largest for ‘"maximize"‘, smallest for ‘"minimize"‘). Goals that tie on the estimated outcome return the same intervention, and the one furthest in the improving direction is taken. The grid is extended in the improving direction until goals stop being reached or affordable, up to 30 times with a warning if that is not enough, and a refinement sweep is then run near the affordability boundary to tighten the estimate.

If no reachable goal is affordable (the budget is below the cost of even the cheapest reachable intervention), the result has ‘feasible = FALSE‘ and ‘NA‘ recommendation fields. If every reachable goal is affordable, the budget does not bind (‘binding = FALSE‘) and the maximum reachable outcome is returned. If [lago_optimization()] fails for every goal of a sweep, for example because the outcome model cannot be fitted, the result has ‘feasible = FALSE‘, the warning and the printed result give the error of the first failed run, and ‘run_error‘ holds it.

Value

An object of class ‘"lago_budget"‘, a list with:

budget The supplied budget.

feasible ‘TRUE‘ if at least one reachable intervention was affordable.

binding ‘TRUE‘ if the budget constrained the choice: a strictly better outcome was reachable but cost more than the budget. ‘FALSE‘ if the best reachable outcome was already affordable.

rec_int The recommended intervention (one value per component), or ‘NA‘ when not ‘feasible‘.

rec_int_cost Its cost (at most ‘budget‘), or ‘NA‘.

est_outcome Its estimated outcome, or ‘NA‘.

outcome_goal The outcome goal that produced the recommendation (equal to ‘est_outcome‘ up to the grid resolution), or ‘NA‘.

frontier A ‘data.frame‘ of the reachable goals searched (the cost/outcome tradeoff curve): ‘outcome_goal‘, ‘rec_int_cost‘, ‘est_outcome‘, ‘affordable‘, ‘status‘.

component_names The intervention component names.

intention The ‘outcome_goal_intention‘ used.

run_error The error of the first failed run when every run of a sweep failed, otherwise ‘NULL‘.

See Also

[lago_optimization()], [lago_sensitivity()]

Other LAGO functions: `get_confidence_set()`, `lago_optimization()`, `lago_report()`, `lago_sensitivity()`, `visualize_cost()`

Examples

```
# How good an outcome can a budget of 100 buy? (mtcars, continuous.)
b <- lago_budget(
  data = mtcars,
  outcome_name = "mpg",
  outcome_type = "continuous",
  glm_family = "gaussian",
  link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal_intention = "maximize",
  budget = 100
)
b
plot(b)

# Reuse a fitted result's call instead of retyping it.
opt <- lago_optimization(
  data = mtcars, outcome_name = "mpg", outcome_type = "continuous",
  glm_family = "gaussian", link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal = 30, outcome_goal_intention = "maximize",
  include_confidence_set = FALSE, quiet = TRUE
)
lago_budget(opt, budget = 100)
```

`lago_optimization` *Run a LAGO optimization*

Description

Fits the outcome model, calculates the recommended interventions based on an outcome goal and/or a power goal, calculates the confidence set for the recommended interventions, and prints the output. This is the main entry point of the package.

Categorical predictors in the outcome model always use treatment contrasts. This includes factors, ordered factors, character columns and logical columns. This coding overrides global contrast options and contrast attributes supplied on the input columns, without changing the input data or global options. The reference is the first retained factor level, the first sorted character value, or FALSE for a logical column. Ordered covariates keep their level order but do not use polynomial contrasts. Numeric predictors keep their values.

Usage

```
lago_optimization(
  data,
  input_data_structure = "individual_level",
  outcome_name,
  outcome_type,
  intervention_components,
  intervention_lower_bounds,
  intervention_upper_bounds,
  outcome_goal = NULL,
  outcome_goal_intention = "maximize",
  power_goal = NULL,
  power_goal_approach = "unconditional",
  num_centers_in_next_stage = NULL,
  patients_per_center_in_next_stage = NULL,
  icc = NULL,
  power_goal_cluster_id = NULL,
  unit_costs = NULL,
  unit_cost_doses = NULL,
  default_cost_fxn_type = "cubic",
  cost_list_of_vectors = NULL,
  glm_family = "default",
  link = "default",
  optimization_method = "numerical",
  confidence_set_alpha = 0.05,
  weights = NULL,
  center_characteristics = NULL,
  center_characteristics_optimization_values = NULL,
  main_components = NULL,
  time_effect_optimization_value = NULL,
  additional_covariates = NULL,
  center_weights_for_outcome_goal = NULL,
  optimization_grid_search_step_size = NULL,
  confidence_set_grid_step_size = NULL,
  include_confidence_set = TRUE,
  include_center_effects = NULL,
  center_effects_optimization_values = NULL,
  include_time_effects = FALSE,
  include_interaction_terms = FALSE,
  prev_recommended_interventions = NULL,
  shrinkage_threshold = 0.25,
  quiet = FALSE
)
```

Arguments

data	A data.frame. The input dataset containing the variables of interest.

`input_data_structure`
: A character string. The data structure of the input data. Must be either "individual_level" or "center_level". Default value without user specification: "individual_level".

`outcome_name`
: A character string. The name of the column in the dataset that represents the outcome of interest.

`outcome_type`
: A character string. Specifies the type of the outcome. Must be either "continuous" for continuous outcomes or "binary" for binary outcomes.

`intervention_components`
: A character vector. The names of the columns in the dataset that represent the intervention components. For example: c("component1", "component2").

`intervention_lower_bounds`
: A numeric vector. Specifies the lower bounds of the intervention components. For example: for a two-component intervention package, lower bounds could be c(0,0).

`intervention_upper_bounds`
: A numeric vector. Specifies the upper bounds of the intervention components. For example: for a two-component intervention package, upper bounds could be c(10,20).

`outcome_goal`
: A numeric value. Specifies the outcome goal, a desired probability or mean value. Default value without user specification: NULL. At least one of outcome_goal or power_goal must be provided. If only a power goal is provided, the optimization target is the outcome level needed to achieve the power goal. If both are provided, the effective outcome goal is the higher of the two (see power_goal).

`outcome_goal_intention`
: A character string. Specifies the direction in which the interventions are expected to move the outcome, relative to the outcome without any intervention. Must be either "maximize" or "minimize". Set it to "maximize" when a larger outcome is desired and the interventions are meant to increase the outcome up to the outcome goal (for example, increasing the probability that a treatment is administered). Set it to "minimize" when a smaller outcome is desired and the interventions are meant to decrease the outcome down to the outcome goal (for example, reducing a mortality rate). This argument only controls the optimization direction; the outcome goal itself is always given as the target outcome value, regardless of the intention. Default value without user specification: "maximize". Note: "minimize" cannot be combined with a power goal, because achieving a power goal requires increasing the outcome.

`power_goal`
: A numeric value. Specifies the power goal, a desired power value (between 0 and 1). Only supported for binary outcomes, and requires a 'group' column (values "treatment"/"control", with both arms present and at least one observed outcome in each) along with num_centers_in_next_stage and patients_per_center_in_next_stage. At least one of outcome_goal or power_goal must be provided. The power goal is converted into the outcome level needed to achieve that power; the effective outcome goal used for optimization is max(power-implied outcome, outcome_goal). When only a power goal is provided, the power-implied outcome is used directly. Default value without user specification: NULL.

power_goal_approach
: A character string. Specifies the approach to achieve the power goal. Must be either "unconditional" or "conditional". Default value without user specification: "unconditional".

num_centers_in_next_stage
: A numeric value. Specifies the number of total centers (treatment + control) in the next stage of the trial. Default value without user specification: NULL.

patients_per_center_in_next_stage
: A numeric value. Specifies the number of patients per center in the next stage of the trial. Default value without user specification: NULL.

icc
: A numeric value in [0, 1), or a length-2 numeric vector c(control, treatment). Specifies the intra-cluster correlation used to inflate the variance of the power-calculation test statistic by a design effect, so that the power goal accounts for within-center clustering. Only used when a power_goal is provided (ignored otherwise). When NULL, the power calculation assumes independent observations (the original behavior); icc = 0 gives the identical result. A larger icc raises the outcome level needed to meet the power goal, so it recommends a stronger intervention. A non-zero icc requires power_goal_cluster_id. Default value without user specification: NULL.

power_goal_cluster_id
: A character string. The name of a column in the data identifying the stage-1 centers (clusters), used to compute the stage-1 design effect for the power calculation. Required when icc is non-zero. The stage-1 cluster size is the variance-appropriate (size-biased) mean sum(m_i^2) / sum(m_i) over that arm's centers, where m_i counts the center's rows with an observed outcome, and each arm must have at least two centers with an observed outcome. Default value without user specification: NULL.

unit_costs
: A numeric vector. Specifies the unit costs for each intervention component. Default value without user specification: NULL.

unit_cost_doses
: A numeric vector or NULL. The dose of each intervention component at which its unit cost in 'unit_costs' was measured, one per unit cost and within the bounds. The cubic cost is then pinned there, so the cost per unit at that dose is the unit cost, and the curve does not depend on the units of the cost or of the dose. Without it the cubic cost only matches each unit cost on average over the bounds, and a warning says so. The linear cost is the unit cost at every dose, so it does not use this. Default value without user specification: NULL.

default_cost_fxn_type
: A character string. Specifies the shape of the default cost function. Must be either "linear" or "cubic". Default value without user specification: "cubic".

cost_list_of_vectors
: A list of numeric vectors. Specifies the cost functions for each intervention component. Each numeric vector in the list contains coefficients of the cost function for one intervention component. For example: list(c(1, 2, 3, 4), c(4, 6), c(5, 4, 3)) represents: - First component: cost = 1 + 2x_1 + 3x_1^2 + 4x_1^3 - Second component: cost = 4 + 6x_2 - Third component: cost = 5 + 4x_3 + 3x_3^2 list(c(0, 2), c(0, 6), c(0, 4)) represents: - First component:

cost = 2x_1 - Second component: cost = 6x_2 - Third component: cost = 4x_3 Default value without user specification: NULL. Please note that either the unit_costs or cost_list_of_vectors must be specified. If both are specified, the cost_list_of_vectors will be used.

`glm_family` A character string. The family of the glm() outcome model. Default value without user specification: "gaussian" for continuous outcomes, "binomial" for binary outcomes.

`link` A character string. The link function of the glm() outcome model. Must be either "logit" or "identity": those are the links whose inverse the estimated outcome and the confidence set are computed with, so they are the only ones that can be reported on the outcome scale the goal is stated on. Default value without user specification: "logit" for binary outcomes and for the "quasibinomial" family, and "identity" otherwise.

`optimization_method`
A character string. Specifies the method used for LAGO optimization. Must be either "numerical" or "grid_search". Default value without user specification: "numerical". - Use "grid_search" if you want to exhaustively test every possible intervention package compositions in LAGO optimization. - Use "numerical" if you want to use gradient-based technique in LAGO optimization. Note: if optimization_grid_search_step_size is provided while this is "numerical", it is automatically switched to "grid_search" (with a message).

`confidence_set_alpha`
A numeric value. The type I error considered in the confidence set calculations. Default value without user specification: 0.05.

`weights` A numeric vector. The weights argument of the glm() outcome model. Continuous confidence covariance uses the actual fitted prior weights as relative observation weights, with zero weights excluded from covariance contributions and the identity residual degrees of freedom. See get_confidence_set() for the covariance formulas and family limits. A data column named weights does not replace this argument. For center level data it is ignored with a warning, since each row is weighted by its center_sample_size.

`center_characteristics`
A character vector. The names of the columns in the dataset that represent the center characteristics. Note, include_center_effects and center_characteristics cannot be used together at the same time. For example: c("characteristic1")

`center_characteristics_optimization_values`
A numeric vector. The fixed values of the center characteristics at which the recommended intervention is computed. Because the outcome model can depend on center characteristics, the recommendation is specific to a center with these characteristic values, i.e. the optimization answers "what is the optimal intervention for a center whose characteristics equal these values?". Must be provided when center_characteristics is used, must be numeric, and must have the same length and order as center_characteristics. For a categorical characteristic with two levels, use 0 for the reference level and 1 for the other level. For a logical characteristic, these values select FALSE and TRUE. Categorical characteristics with more than two levels are not supported. Recode them as separate numeric indicator columns. For example, with center_characteristics

= c("birth_volume_100"), setting center_characteristics_optimization_values = c(1.75) computes the recommended intervention for a center with birth_volume_100 = 1.75.

`main_components`
: A character vector. Specifies the main intervention components in the presence of interaction terms. For example: c("component1", "component2").

`time_effect_optimization_value`
: A numeric value. The value of the time effect that will be used for LAGO optimization.

`additional_covariates`
: A character vector. The names of the columns in the dataset that represent additional covariates that need to be included in the outcome model. This includes interaction terms or any other additional covariates. For recommendations and confidence output, numeric covariates are held at 0 and categorical covariates are held at their reference level. Set a factor's level order before calling this function to choose its reference level. Categorical additional covariates can have more than two levels. For example: c("component2xcomponent4").

`center_weights_for_outcome_goal`
: A numeric vector. Specifies the weights that will be used for calculating recommended interventions that satisfy the outcome goal for an (weighted) average center. The weights need to sum up to 1, and must all be non-negative and finite. A weight of 0 is allowed and excludes that center from the average. A center with no row where every model variable and the row's glm weight (center_sample_size for center level data) are observed is left out of the fit, so its weight must be 0. The weights follow the order of the centers: the levels of a factor column once unused levels are dropped (an explicit NA level included), and the sorted values otherwise. Default value without user specification: For each center the model fitted, calculate what percentage its sample size (all of its rows, or the sum of its center_sample_size for center level data) is of the total over those centers, and use that percentage as the center's weight. A center the model could not fit is left out, with a warning. For center level data, a center with no center_sample_size at all is refused instead.

`optimization_grid_search_step_size`
: A numeric vector. Specifies the step size of the grid search algorithm used in LAGO optimization. This argument is only used by the "grid_search" method; if it is provided while optimization_method is "numerical", optimization_method is automatically switched to "grid_search" (with a message). Default value without user specification: 1/20 of the range for each intervention component.

`confidence_set_grid_step_size`
: A numeric vector. Specifies the step size of the grid search algorithm used in the confidence set calculation. Default value without user specification: 1/20 of the range for each intervention component.

`include_confidence_set`
: A boolean. Specifies whether the confidence set should be calculated along with the recommended interventions. Default value without user specification: TRUE.

include_center_effects
: A boolean. Specifies whether the fixed effects should be included in the outcome model. If set to TRUE, please make sure that the input data has a 'center' column to identify the centers. An ordered factor is converted to an unordered one, so each center gets its own effect, and at least two centers need a row the outcome model can use. Note, include_center_effects and center_characteristics cannot be used together at the same time. Default value without user specification (NULL): FALSE for individual level data, TRUE for center level data.

center_effects_optimization_values
: A character string. The name of the center of interest (one of the values of the 'center' column) that will be used for center-specific LAGO optimization. This is only used when include_center_effects is set to TRUE. If not specified, the LAGO optimization will be carried out for a weighted average of the centers the model fitted (see center_weights_for_outcome_goal). The center must have a row where every model variable and the row's glm weight (center_sample_size for center level data) are observed. To select an explicit NA center level (as from `addNA()`), pass `NA_character_`. The string `"NA"` selects only a center whose label is the text NA. A center labeled with the text NA and an explicit NA level cannot be used together, since the model gives both the same coefficient name, so rename one of them.

include_time_effects
: A boolean. Specifies whether the fixed time effects should be included in the outcome model. If set to TRUE, please make sure that the input data has a 'period' column to identify the time periods. An ordered factor is converted to an unordered one, so each period gets its own effect, and at least two periods need a row the outcome model can use. Default value without user specification: FALSE.

include_interaction_terms
: A boolean. Specifies whether there are interaction terms in the intervention components. Please make sure the interaction terms are included in "intervention_components", and they follow the correct naming scheme: "component1:component3". Default value without user specification: FALSE.

prev_recommended_interventions
: A numeric vector. The recommended interventions from a previous stage of the LAGO optimization. If provided, this will be used in the shrinking method (see Section 5.1 of the Supplementary material of Nevo et al. (2021) for details).

shrinkage_threshold
: A numeric value between 0 and 1. Not used, and kept so that existing calls still run. When the outcome goal is not estimated to be reachable within the bounds, the shrinkage method from Section 5.1 of the Supplementary Materials of Nevo et al. (2021) is always applied, whatever this value is. Default value without user specification: 0.25.

quiet
: A boolean. If TRUE, suppresses the progress messages, the paced delays between them, and the final printed output, which speeds up the call substantially (the default paced output adds about two seconds of artificial delay). The returned value is identical either way. Genuine warnings about the data or model fit are still shown. Useful for programmatic or repeated calls (for example, simulations). Default value without user specification: FALSE.

Value

List(recommended interventions, associated cost for the interventions, estimated outcome mean/probability for the intervention group, confidence set percentage (at level 1 - confidence_set_alpha), confidence set, overall intervention test results (test_results), a list with the test statistic and p-value; NULL unless a valid 'group' column is supplied. shrinking_method_used, TRUE when the outcome goal could not be reached and the shrinking method gave the recommendation, in which case no confidence set is computed. The confidence set fields are only present when include_confidence_set = TRUE.) The result also carries its own call arguments in the attribute '"lago_call_args"' (not a '$'-accessible field), so it can be passed directly to [lago_sensitivity()] to sweep one input without retyping the call.

See Also

[get_confidence_set()], [visualize_cost()], [lago_report()]

Other LAGO functions: get_confidence_set(), lago_budget(), lago_report(), lago_sensitivity(), visualize_cost()

Examples

```
# Basic case showing how to carry out the optimization with
# a built-in data set. The confidence set is skipped here to keep this
# example fast; the \donttest examples below show a full run with one.
lago_optimization(
  data = infert,
  outcome_name = "case",
  outcome_type = "binary",
  glm_family = "binomial",
  intervention_components = c("age", "parity"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(50, 10),
  cost_list_of_vectors = list(c(0, 4), c(0, 1)),
  outcome_goal = 0.5,
  outcome_goal_intention = "maximize",
  include_confidence_set = FALSE
)

# Fuller examples, including the confidence-set grid search, which is
# inherently slower. Wrapped in \donttest{} so they are still checkable
# but do not run on every automated check.

lago_optimization(
  data = BB_data,
  outcome_name = "pp3_oxytocin_mother",
  outcome_type = "binary",
  glm_family = "binomial",
  intervention_components = c("coaching_updt", "launch_duration"),
  center_characteristics = c("birth_volume_100"),
  center_characteristics_optimization_values = 1.75,
  intervention_lower_bounds = c(1, 1),
  intervention_upper_bounds = c(40, 5),
  cost_list_of_vectors = list(c(0, 1700), c(0, 8000)),
```

```
  outcome_goal = 0.85,
  outcome_goal_intention = "maximize",
  confidence_set_grid_step_size = c(1, 1)
)

lago_optimization(
  data = BB_proportions,
  outcome_name = "EBP_proportions",
  outcome_type = "continuous",
  glm_family = "quasibinomial",
  link = "logit",
  intervention_components = c("coaching_updt", "launch_duration"),
  center_characteristics = c("birth_volume_100"),
  center_characteristics_optimization_values = 1.75,
  intervention_lower_bounds = c(1, 1),
  intervention_upper_bounds = c(40, 5),
  cost_list_of_vectors = list(c(0, 1700), c(0, 8000)),
  outcome_goal = 0.85,
  outcome_goal_intention = "maximize",
  confidence_set_grid_step_size = c(1, 1)
)
```

`lago_report` *Generate an HTML report for a LAGO optimization result*

Description

Renders a self-contained, interactive HTML report for the object returned by [lago_optimization()]. The report knits a bundled parameterized R Markdown template and lays out the same sections, in the same order and with the same labels, as the console methods ([print.lago()] / [summary.lago()]): the recommended intervention, its cost, the estimated outcome (with its confidence interval, 95% by default) and goal, the power goal (when present), the confidence set (size, cost IQR, and first rows), the overall intervention-effect test (when present), and the fitted outcome-model coefficient table. A ‘sessionInfo()‘ footer records the environment the report was produced in.

No new statistics are computed; every value is taken verbatim from an existing field on ‘x‘ and formatted exactly as the console output formats it.

Usage

```
lago_report(
  x,
  output_file = NULL,
  title = "LAGO optimization report",
  open = FALSE,
  ...
)
```

Arguments

x	A "lago" object returned by [lago_optimization()].
output_file	A character string. The path to write the HTML report to. If ‘NULL‘ (the default), a temporary file with a ‘.html‘ extension is used.
title	A character string. The report title. Default value without user specification: "LAGO optimization report".
open	A boolean. If ‘TRUE‘, opens the rendered report in a browser via [utils::browseURL()]. Default value without user specification: ‘FALSE‘.
...	Additional arguments passed to [rmarkdown::render()].

Details

The report is interactive and drawn entirely client-side with the bundled D3 v7 library (‘inst/js/d3.v7.min.js‘) plus ‘inst/js/lago-report.js‘, both inlined into the HTML so the output is a single self-contained, offline file with no external asset references (no CDN, no Shiny, no additional widget frameworks). It adds two interactive views:

- an interactive confidence-set plot: a hover-enabled scatter for a two-component intervention (with the recommended intervention highlighted) or a one-dimensional strip for a single component. For 3+ components, or when the interactive view is not applicable, it falls back to the static [plot.lago()] figure; when no confidence set is available the section is omitted.
- interactive cost curves: for each intervention component, its total-cost curve and marginal (per-unit) cost curve, each with a hover read-out, drawn from the result's cost functions and bounds.

Rendering requires the **rmarkdown**, **knitr**, and **jsonlite** packages (all listed under Suggests). If they are not installed, an informative error is raised.

Value

The path to the rendered HTML file, invisibly.

See Also

[lago_optimization()]

Other LAGO functions: `get_confidence_set()`, `lago_budget()`, `lago_optimization()`, `lago_sensitivity()`, `visualize_cost()`

Examples

```
# lago_report() renders an HTML file with rmarkdown, so it is wrapped in
# \dontrun: it writes to disk and invokes pandoc, and running a document
# render inside an in-process documentation build (e.g. pkgdown) can deadlock.
## Not run:
result <- lago_optimization(
  data = BB_data,
  outcome_name = "pp3_oxytocin_mother",
  outcome_type = "binary",
```

```
  glm_family = "binomial",
  intervention_components = c("coaching_updt", "launch_duration"),
  center_characteristics = c("birth_volume_100"),
  center_characteristics_optimization_values = 1.75,
  intervention_lower_bounds = c(1, 1),
  intervention_upper_bounds = c(40, 5),
  cost_list_of_vectors = list(c(0, 1700), c(0, 8000)),
  outcome_goal = 0.85,
  outcome_goal_intention = "maximize",
  confidence_set_grid_step_size = c(1, 1),
  quiet = TRUE
)
report_path <- lago_report(result)

## End(Not run)
```

`lago_sensitivity` *Sensitivity analysis for a LAGO recommendation*

Description

Trial designers often are unsure about some of the inputs to [lago_optimization()], chiefly the outcome goal and the assumed intervention costs. ‘lago_sensitivity()‘ answers "how much does the recommendation move if that input is different?" by re-running [lago_optimization()] across a sweep of one input and reporting how the recommended intervention, its cost, and the estimated outcome change. This turns a single point recommendation into a robustness picture: a stringency curve for the outcome goal, or a straight line for a uniform cost rescaling.

The confidence set is the slow part of [lago_optimization()] and is not needed to see how a recommendation moves, so every run is forced to ‘include_confidence_set = FALSE‘ for speed. A single run failing does not abort the sweep: its outputs are recorded as ‘NA‘ and a single warning names the failed values afterwards.

Usage

```
lago_sensitivity(object = NULL, ..., parameter, values, quiet = TRUE)
```

Arguments

`object`	An optional ‘lago‘ result from [lago_optimization()]. When supplied, the baseline optimization arguments are taken from the call it carries (so the whole call need not be retyped), and any arguments in ‘...‘ override those stored values. When ‘NULL‘ (the default), the baseline arguments come from ‘...‘. Passing an object that is not a ‘lago‘ result is an error, as is passing a ‘lago‘ result from an older version of the package that did not record its call arguments.
`...`	The baseline [lago_optimization()] arguments (the user’s own optimization call). They are forwarded unchanged to every run. When ‘object‘ is not supplied, these are the baseline arguments and the user typically copies their ‘lago_optimization(...)‘

call in and simply adds ‘parameter‘ and ‘values‘. When ‘object‘ is supplied, anything here overrides the arguments stored on the object. ‘include_confidence_set‘ and ‘quiet‘ supplied here are overridden (see Details).

`parameter` A single character string naming what to vary. Two modes:

1. The name of a scalar numeric argument of [lago_optimization()] that affects the recommendation (for example ‘"outcome_goal"‘, ‘"power_goal"‘, or ‘"icc"‘). Each run overrides that argument with one element of ‘values‘. Vector-valued arguments such as ‘intervention_lower_bounds‘, and confidence-set-only arguments such as ‘confidence_set_alpha‘ (the confidence set is not computed during a sweep), are rejected.
2. The special string ‘"cost_multiplier"‘: each run multiplies every coefficient of every vector in the baseline ‘cost_list_of_vectors‘ by one element of ‘values‘, so ‘values = c(0.8, 1, 1.2)‘ evaluates the costs at 80%, 100%, and 120%. This mode requires ‘cost_list_of_vectors‘ in ‘...‘ and all ‘values‘ must be positive.

`values` A non-empty numeric vector, all finite. One run per element.

`quiet` A boolean forwarded to [lago_optimization()]. Defaults to ‘TRUE‘ so the sweep is not noisy. Genuine warnings about the data or model fit from each run are still shown.

Details

Each run builds the modified argument list from ‘...‘, forces ‘include_confidence_set = FALSE‘, sets ‘quiet = quiet‘, and calls [lago_optimization()] inside ‘tryCatch()‘. A run that errors contributes a row of ‘NA‘ outputs with a non-‘"ok"‘ status instead of stopping the sweep.

Value

An object of class ‘"lago_sensitivity"‘, which is a ‘data.frame‘ with one row per element of ‘values‘ and columns:

value The swept value for that run.

<component> One numeric column per intervention component, named by the component, holding its recommended value for that run.

rec_int_cost The cost of the recommended intervention.

est_outcome_goal The estimated outcome at the recommendation.

status ‘"ok"‘ for a successful run, otherwise ‘"error"‘.

The object carries attributes ‘parameter‘ (the swept string), ‘component_names‘ (the component column names), ‘baseline‘ (the full [lago_optimization()] result at the baseline value, i.e. multiplier 1 for ‘"cost_multiplier"‘ or the value supplied in ‘...‘ for a named parameter, or ‘NULL‘ if not present), and, when any run failed, ‘error_messages‘ (the error text named by the failed values).

See Also

[lago_optimization()]

Other LAGO functions: `get_confidence_set()`, `lago_budget()`, `lago_optimization()`, `lago_report()`, `visualize_cost()`

Examples

```
# Recommended flow: fit once, then sweep the fitted result. Passing the
# `lago` result reuses its call, so there is no need to retype the
# arguments. Each run is a separate optimization, so this is wrapped in
# \donttest to keep automated checks fast; the confidence set is off
# internally so the sweep still runs quickly.
opt <- lago_optimization(
  data = mtcars,
  outcome_name = "mpg",
  outcome_type = "continuous",
  glm_family = "gaussian",
  link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal = 30,
  outcome_goal_intention = "maximize",
  include_confidence_set = FALSE,
  quiet = TRUE
)
sens <- lago_sensitivity(
  opt,
  parameter = "outcome_goal",
  values = c(30, 35, 40)
)
sens
plot(sens)

# The same sweep spelled out directly (no fitted object). This is the
# equivalent long form and gives the same result.
sens2 <- lago_sensitivity(
  data = mtcars,
  outcome_name = "mpg",
  outcome_type = "continuous",
  glm_family = "gaussian",
  link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal_intention = "maximize",
  parameter = "outcome_goal",
  values = c(30, 35, 40)
)

# How sensitive is it to the assumed costs? A uniform rescaling never changes
# which intervention is cheapest, so the recommendation is unchanged and the
# cost scales linearly with the multiplier. `...` overrides on top of the
# fitted object are used here to switch the swept parameter.
cost_sens <- lago_sensitivity(
  opt,
```

```
  parameter = "cost_multiplier",
  values = c(0.8, 1, 1.2)
)
cost_sens
```

`main_pulesa_data` *PULESA hypertension-care data (main model)*

Description

Clinic-level data from the PULESA study, a cluster-randomized trial of a multi-component intervention to improve hypertension care. Each row is one clinic in one time period. This is the version used to fit the "main" LAGO outcome model, whose intervention components are access to blood-pressure machines, access to medicines, delivery mode, hypertension training, performance improvement, and remote monitoring.

Usage

```
main_pulesa_data
```

Format

A data frame with 192 rows and 15 variables:

Clinic Clinic (cluster) identifier

Period Study time period

AccessBPMachines Intervention component: access to blood-pressure machines

AccessMedicines Intervention component: access to medicines

DeliveryA Intervention component: care delivery mode A

DeliveryB Intervention component: care delivery mode B

HypertensionTraining Intervention component: hypertension training

MI_DeliveryB Interaction term involving delivery mode B

MI_PerformanceImprovement Interaction term involving performance improvement

MI_RemoteMonitoring Interaction term involving remote monitoring

PerformanceImprovement Intervention component: performance improvement

Proportions Outcome: proportion of successes at the clinic-period

RemoteMonitoring Intervention component: remote monitoring

Success Outcome: number of successes at the clinic-period

Total_visit Number of visits at the clinic-period

Examples

```
data(main_pulesa_data)
head(main_pulesa_data)
```

`new_pulesa_data` *PULESA hypertension-care data (new model)*

Description

Clinic-level data from the PULESA study, a cluster-randomized trial of a multi-component intervention to improve hypertension care. Each row is one clinic in one time period. This is the version used to fit the "new" LAGO outcome model, which uses the same clinics and periods as `main_pulesa_data` but a different specification of the intervention (the `W1`–`W8` components and their interaction terms).

Usage

```
new_pulesa_data
```

Format

A data frame with 192 rows and 22 variables:

Clinic Clinic (cluster) identifier

Period Study time period

AccessBPMachines Intervention component: access to blood-pressure machines

AccessMedicines Intervention component: access to medicines

DeliveryA Intervention component: care delivery mode A

DeliveryB Intervention component: care delivery mode B

HypertensionTraining Intervention component: hypertension training

MI_W5 Interaction term involving component W5

MI_W7 Interaction term involving component W7

MI_W8 Interaction term involving component W8

PerformanceImprovement Intervention component: performance improvement

RemoteMonitoring Intervention component: remote monitoring

Success Outcome: number of successes at the clinic-period

Total_visit Number of visits at the clinic-period

W1 Intervention component W1

W2 Intervention component W2

W3 Intervention component W3

W4 Intervention component W4

W5 Intervention component W5

W6 Intervention component W6

W7 Intervention component W7

W8 Intervention component W8

Examples

```
data(new_pulesa_data)
head(new_pulesa_data)
```

`plot.lago` *Plot a LAGO optimization result*

Description

Visualizes the confidence set, 95% by default. For a two-component intervention it plots the grid points in the confidence set with the recommended intervention highlighted; for a single component it plots the confidence interval bounds against the dose. A non-empty confidence set is required. ‘result$cs‘ can be NULL even with ‘include_confidence_set = TRUE‘ (its default), when no confidence set was found for the outcome goal or the shrinking method was used, and then plot() returns invisibly with a message rather than erroring.

Usage

```
## S3 method for class 'lago'
plot(x, ...)
```

Arguments

`x`	A "lago" object returned by [lago_optimization()].
`...`	Ignored.

Value

A ggplot object (invisibly) when a plot is produced, otherwise ‘NULL‘ invisibly.

Examples

```
# A plot needs a non-empty confidence set: plot() returns invisibly with a
# message when result$cs is NULL, which can happen even with
# include_confidence_set = TRUE (its default) if no confidence set was found
# for the outcome goal, or if the shrinking method was used.
# The lower bounds start at 1 while the data also contains 0s, so the
# optimizer warns about that; the warning is expected here.
result <- lago_optimization(
  data = BB_data,
  outcome_name = "pp3_oxytocin_mother",
  outcome_type = "binary",
  glm_family = "binomial",
  intervention_components = c("coaching_updt", "launch_duration"),
  center_characteristics = c("birth_volume_100"),
  center_characteristics_optimization_values = 1.75,
  intervention_lower_bounds = c(1, 1),
  intervention_upper_bounds = c(40, 5),
```

```
  cost_list_of_vectors = list(c(0, 1700), c(0, 8000)),
  outcome_goal = 0.85,
  outcome_goal_intention = "maximize",
  include_confidence_set = TRUE,
  confidence_set_grid_step_size = c(1, 1),
  quiet = TRUE
)

# Two components: the confidence set grid with the recommended
# intervention marked.
plot(result)
```

`plot.lago_budget` *Plot a budget-constrained LAGO result*

Description

Draws the cost/outcome frontier the search traced out (estimated outcome against recommended cost across the swept goals), with a vertical line at the budget and the chosen recommendation highlighted.

Usage

```
## S3 method for class 'lago_budget'
plot(x, ...)
```

Arguments

`x`	A ‘"lago_budget"‘ object from [lago_budget()].
`...`	Ignored.

Value

A [ggplot2][ggplot2::ggplot] object, or ‘NULL‘ invisibly when there is nothing reachable to plot.

Examples

```
b <- lago_budget(
  data = mtcars, outcome_name = "mpg", outcome_type = "continuous",
  glm_family = "gaussian", link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal_intention = "maximize", budget = 100
)
plot(b)
```

`plot.lago_sensitivity` *Plot a LAGO sensitivity analysis*

Description

Draws the sensitivity curve with [ggplot2][ggplot2::ggplot]. By default (‘show = "cost"‘) it plots the recommended cost against the swept value as a line with points: for an ‘"outcome_goal"‘ sweep this is the cost-of-stringency curve, and for a ‘"cost_multiplier"‘ sweep it is a straight line. With ‘show = "components"‘ it instead plots the recommended value of each intervention component against the swept value, one coloured line per component. Rows with ‘NA‘ outputs are dropped with a message.

Usage

```
## S3 method for class 'lago_sensitivity'
plot(x, show = c("cost", "components"), ...)
```

Arguments

`x`	A ‘"lago_sensitivity"‘ object from [lago_sensitivity()].
`show`	One of ‘"cost"‘ (default, plot ‘rec_int_cost‘) or ‘"components"‘ (plot the per-component recommended values).
`...`	Ignored.

Value

A [ggplot2][ggplot2::ggplot] object, or ‘NULL‘ invisibly when ‘show = "components"‘ and no components are recorded.

Examples

```
sens <- lago_sensitivity(
  data = mtcars, outcome_name = "mpg", outcome_type = "continuous",
  glm_family = "gaussian", link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal_intention = "maximize",
  parameter = "outcome_goal", values = c(30, 35, 40)
)
plot(sens)
plot(sens, show = "components")
```

print.lago *Print a LAGO optimization result*

Description

Full console display of the object returned by [lago_optimization()], rendered with boxed, colour-accented [cli][cli::cli] sections: an inputs recap (data dimensions, outcome, intervention components, model family/link and fixed effects, goals, costs and bounds), the fitted outcome-model coefficient table, the overall intervention-effect test, the recommended intervention with its cost and the estimated outcome (and its confidence interval, 95% by default), and the confidence set (size, cost IQR, and first rows). Everything is shown on the console so results can be read without further calls. [summary.lago()] renders the same output.

Usage

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

Arguments

x	A "lago" object returned by [lago_optimization()].
...	Ignored.

Value

'x', invisibly.

Examples

```
# lago_optimization() already prints the result, so quiet = TRUE avoids
# rendering it twice here. The lower bounds start at 1 while the data also
# contains 0s, so the optimizer warns about that; the warning is expected.
result <- lago_optimization(
  data = BB_data,
  outcome_name = "pp3_oxytocin_mother",
  outcome_type = "binary",
  glm_family = "binomial",
  intervention_components = c("coaching_updt", "launch_duration"),
  center_characteristics = c("birth_volume_100"),
  center_characteristics_optimization_values = 1.75,
  intervention_lower_bounds = c(1, 1),
  intervention_upper_bounds = c(40, 5),
  cost_list_of_vectors = list(c(0, 1700), c(0, 8000)),
  outcome_goal = 0.85,
  outcome_goal_intention = "maximize",
  include_confidence_set = TRUE,
  confidence_set_grid_step_size = c(1, 1),
  quiet = TRUE
)
```

```
print(result)
```

`print.lago_budget` *Print a budget-constrained LAGO result*

Description

Prints a short header with the budget and whether it binds, then the recommended intervention, its cost and estimated outcome (or a note that no intervention fits the budget).

Usage

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

Arguments

`x`	A ‘"lago_budget"‘ object from [lago_budget()].
`...`	Ignored.

Value

‘x‘, invisibly.

Examples

```
b <- lago_budget(
  data = mtcars, outcome_name = "mpg", outcome_type = "continuous",
  glm_family = "gaussian", link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal_intention = "maximize", budget = 100
)
print(b)
```

print.lago_sensitivity *Print a LAGO sensitivity analysis*

Description

Prints a short header naming the varied parameter, the number of runs and how many failed, then the sensitivity ‘data.frame‘, then a one-line summary of how ‘rec_int_cost‘ ranges over the sweep.

Usage

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

Arguments

x	A ‘"lago_sensitivity"‘ object from [lago_sensitivity()].
...	Ignored.

Value

‘x‘, invisibly.

Examples

```
sens <- lago_sensitivity(
  data = mtcars, outcome_name = "mpg", outcome_type = "continuous",
  glm_family = "gaussian", link = "identity",
  intervention_components = c("gear", "qsec"),
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 350),
  cost_list_of_vectors = list(c(0, 4), c(4, 6)),
  outcome_goal_intention = "maximize",
  parameter = "outcome_goal", values = c(30, 35, 40)
)
print(sens)
```

summary.lago *Summarize a LAGO optimization result*

Description

Renders the same full console display as [print.lago()]: the inputs recap, outcome-model coefficient table, overall intervention-effect test, recommended intervention (with cost and the estimated-outcome CI), and the confidence set. Provided so ‘summary()‘ works as expected on a "lago" object.

Usage

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

Arguments

object	A "lago" object returned by [lago_optimization()].
...	Ignored.

Value

‘object‘, invisibly.

Examples

```
# summary() currently renders exactly the same output as print(). The lower
# bounds start at 1 while the data also contains 0s, so the optimizer warns
# about that; the warning is expected here.
result <- lago_optimization(
  data = BB_data,
  outcome_name = "pp3_oxytocin_mother",
  outcome_type = "binary",
  glm_family = "binomial",
  intervention_components = c("coaching_updt", "launch_duration"),
  center_characteristics = c("birth_volume_100"),
  center_characteristics_optimization_values = 1.75,
  intervention_lower_bounds = c(1, 1),
  intervention_upper_bounds = c(40, 5),
  cost_list_of_vectors = list(c(0, 1700), c(0, 8000)),
  outcome_goal = 0.85,
  outcome_goal_intention = "maximize",
  include_confidence_set = TRUE,
  confidence_set_grid_step_size = c(1, 1),
  quiet = TRUE
)

summary(result)
```

`visualize_cost` *Visualize and choose intervention cost functions*

Description

Visualize the cost functions for the intervention components. This function creates a Shiny app that allows the user to adjust the coefficients of the cost functions for each intervention component and visualize the resulting total cost function and its derivative. The initial coefficients are calculated based on the unit costs, the cost function type (linear or cubic), and the lower and upper bounds. A "Cost function form" toggle at the top switches all components between the linear (2 coefficients) and cubic (a degree-4 total cost, 5 coefficients) forms, resetting each to that form's initial coefficients. The user can adjust the coefficients using sliders and reset them to their initial values. Each slider has a default range scaled to its coefficient's magnitude (derived from the unit costs and bounds), and the user can set a custom range for any slider using its "Range min" / "Range max" inputs. The total cost curve can also be reshaped directly: it carries a draggable handle at each of several points along the curve, and dragging one refits the cost function through the new set of points and updates the sliders. The app also displays the current coefficient vector for each component. The user can copy the final coefficient list for use in the optimization function lago_optimization().

Usage

```
visualize_cost(
  component_names,
  unit_costs,
  default_cost_fxn_type,
  intervention_lower_bounds,
  intervention_upper_bounds
)
```

Arguments

`component_names`
: A character vector of the names of the intervention components.

`unit_costs`
: A numeric vector of the unit costs for each intervention component.

`default_cost_fxn_type`
: A character string specifying the cost function type the app opens with. Must be either "linear" or "cubic". The user can switch forms in the app with the "Cost function form" toggle.

`intervention_lower_bounds`
: A numeric vector of the lower bounds for each intervention component.

`intervention_upper_bounds`
: A numeric vector of the upper bounds for each intervention component.

Details

When the app is closed with the "Return list to R & close" button, the coefficient list is also stored in the `lago_cost_list` option (overwriting any existing value) and a message reports this, so the list is available even when the app was launched with a bare `visualize_cost(...)` call rather than `cost_list <-visualize_cost(...)`. Retrieve it with `getOption("lago_cost_list")`. Closing the browser tab instead of using the button does not save the list.

Value

Invisibly, the cost-function coefficient list (a list of numeric vectors, one per intervention component) as it stood when the app was closed, suitable for passing to `lago_optimization(cost_list_of_vectors = ...)`. The same list can be copied from within the app.

See Also

[lago_optimization()]

Other LAGO functions: `get_confidence_set()`, `lago_budget()`, `lago_optimization()`, `lago_report()`, `lago_sensitivity()`

Examples

```
## Not run:
visualize_cost(
  component_names = c("Component 1", "Component 2"),
  unit_costs = c(0.5, 1),
  default_cost_fxn_type = "linear",
  intervention_lower_bounds = c(0, 0),
  intervention_upper_bounds = c(10, 10)
)

## End(Not run)
```

Index

∗ **LAGO functions**
lago_budget, 6
lago_optimization, 8
lago_report, 16
lago_sensitivity, 18
visualize_cost, 30
∗ **datasets**
BB_data, 2
BB_proportions, 4
main_pulesa_data, 21
new_pulesa_data, 22

BB_data, 2
BB_proportions, 4

get_confidence_set(), *7*, *15*, *17*, *19*, *31*

lago_app, 4
lago_budget, 6
lago_budget(), *15*, *17*, *19*, *31*
lago_optimization, 8
lago_optimization(), *7*, *17*, *19*, *31*
lago_report, 16
lago_report(), *7*, *15*, *19*, *31*
lago_sensitivity, 18
lago_sensitivity(), *7*, *15*, *17*, *31*

main_pulesa_data, 21, *22*

new_pulesa_data, 22

plot.lago, 23
plot.lago_budget, 24
plot.lago_sensitivity, 25
print.lago, 26
print.lago_budget, 27
print.lago_sensitivity, 28

summary.lago, 29

visualize_cost, 30
visualize_cost(), *7*, *15*, *17*, *19*