Package: whatifbandit (via r-universe)

June 1, 2026

Title Analyzing Randomized Experiments as Multi-Arm Bandits

Version 0.3.0.9001

Description Simulates the results of completed randomized controlled trials, as if they had been conducted as adaptive Multi-Arm Bandit (MAB) trials instead. Augmented inverse probability weighted estimation (AIPW), outlined by Hadad et al. (2021) <doi:10.1073/pnas.2014602118>, is used to robustly estimate the probability of success for each treatment arm under the adaptive design. Provides customization options to simulate perfect/imperfect information, stationary/non-stationary bandits, blocked treatment assignments, along with control augmentation, and other hybrid strategies for assigning treatment arms. The methods used in simulation were inspired by Offer-Westort et al. (2021) <doi:10.1111/ajps.12597>.

License GPL (>= 3)

URL https://github.com/Noch05/whatifbandit

BugReports https://github.com/Noch05/whatifbandit/issues

Depends R (>= 4.1.0)

Imports bandit, data.table, dplyr, furrr, ggplot2, lubridate, purrr, randomizr, rlang, tibble, tidyr

Suggests future, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev

Repository https://noch05.r-universe.dev

Date/Publication 2026-02-28 23:58:13 UTC

RemoteUrl https://github.com/noch05/whatifbandit

RemoteRef HEAD

RemoteSha 6e7cf0c1d8f4297c561bfa3733451e5178c8eaed

Contents

`generate_rct.bernoulli` *Generate a Bernoulli RCT dataset*

Description

Simulates a randomized controlled trial with Bernoulli outcomes. Supports complete, block, cluster, and block-and-cluster randomized assignment, optional assignment dates, and a user-supplied time-to-event model for successful observations.

Usage

```
generate_rct.bernoulli(
  n,
  p,
  dt = FALSE,
  blocks = NULL,
  clusters = NULL,
  dates_of_assignment = NULL,
  time_model = NULL,
  ...
)
```

Arguments

`n` A positive integer. Total number of units to simulate.

`p` The true probabilities of success for each treatment arm. Can be:

Numeric vector A named vector where `names(p)` are the treatment labels, e.g. `c(T1 = 0.2, T2 = 0.4)`. Used when there are no blocks or clusters.

Named list of vectors A named list where `names(p)` are the treatment labels and each element is a named vector of per-block or per-cluster success probabilities, e.g. `list(T1 = c(B1 = 0.2, B2 = 0.4), T2 = c(B1 = 0.3, B2 = 0.5))`. Probabilities are accessed as `p[[treatment]][block]`.

Nested named list A doubly-nested named list used when both blocks and clusters are present. The outer names are treatment labels, the inner names are block labels, and each inner most vector is a named vector of per-cluster success probabilities. e.g. `list(T1 = list(B1 = c(C1 = 0.2, C2 = 0.4), B2 = c(C1 = 0.7, C2 =` Accessed as `p[[treatment]][[block]][cluster]`.

All probability values must be between 0 and 1.

`dt` Logical. If `TRUE` returns a `data.table::data.table()`; otherwise returns a `tibble::tibble()`. Default `FALSE`.

`blocks` A named numeric vector of block membership probabilities (must sum to 1), where `names(blocks)` are the block labels. Units are assigned to blocks via `randomizr::complete_ra()`. Pass `NULL` (default) for no blocking.

`clusters` Cluster membership probabilities. Can be:

Numeric vector A named vector where `names(clusters)` are the cluster labels e. g. `C(C1 = 0.4, C2 = 0.6)`. Used when there is not blocking.

Named list of vectors A named list where `names(clusters)` are block labels, and each element is a named vector of per-block cluster proportions, e.g. `list(B1 = c(C1 = 0.4, C=0.6)), B2 = c(C3 = 0.2, C4 = 0.8)` Clusters are accessed as `clusters[[block]][cluster]`. Insided each block, cluster proportions must sum to 1, and the same cluster cannot appear in multiple blocks.

where `names(clusters)` are the cluster labels. Units are assigned to clusters via `randomizr::complete_ra()`. Pass `NULL` (default) for no clustering.

`dates_of_assignment`

An optional vector of dates representing when units are assigned. If shorter than `n` it is recycled and sorted. If NULL‘ (default) no assignment dates are recorded.

`time_model` An optional function with signature `function(n, treatments, success, blocks = NULL, clusters =` that returns a vector of lubridate::period objects to add to `dates_of_assignment` to produce `success_date`. Only used when`dates_of_assignment` is also supplied. Default `NULL`.

`...` Additional arguments forwarded to `time_model`.

Value

A `tibble::tibble()` or `data.table::data.table()` with columns:

`id` Integer unit identifier.

`treatment` Treatment arm assigned to each unit.

`success` Binary outcome (0/1) drawn from a Bernoulli distribution.

`assignment_date` Date of assignment (if `dates_of_assignment` supplied).

`success_date` Date of outcome (if `dates_of_assignment` and `time_model` supplied).

Examples

```
# 4-Arm Design;

generate_rct.bernoulli(n = 100, p = c(0.3, 0.7, 0.5, 0.4))

# Blocked and Clustered with random probabilities

generate_rct.bernoulli(
  n = 100,
  p = list(
    Control = list(
      B1 = c(C1 = 0.6, C2 = 0.3),
      B2 = c(C3 = 0.2, C4 = 0.5)
    ),
    T1 = list(
      B1 = c(C1 = 0.8, C2 = 0.9),
      B2 = c(C3 = 0.2, C4 = 0.4)
    ),
    T2 = list(
      B1 = c(C1 = 0.3, C2 = 0.3),
      B2 = c(C3 = 0.5, C3 = 0.5)
    )
  ),
  blocks = c(B1 = 0.3, B2 = 0.7),
  clusters = list(B1 = c(C1 = 0.3, C2 = 0.7), B2 = c(C3 = 0.2, C4 = 0.8))
)

# Modelling Success Dates

time_model <- function(n, treatments, success, blocks, clusters = NULL) {
  # Specific model for each treatment and block
  time <- data.table::fcase(
    treatments == "T1" & blocks == "B1" , rexp(n, rate = 5)                ,
    treatments == "T1" & blocks == "B2" , rexp(n, rate = 10)               ,
    treatments == "T2" & blocks == "B1" , rgamma(n, shape = 2, rate = 2)   ,
    treatments == "T2" & blocks == "B2" , rweibull(n, shape = 3, scale = 4)
  )

  time <- ifelse(success == 1, round(time) * lubridate::days(1), NA)
  return(time)
}
# Using `single_mab_simulation()`

set.seed(100)
result <- generate_rct.bernoulli(
  n = 10000,
```

```
  p = list(
    T1 = c(B1 = 0.5, B2 = 0.3, B3 = 0.2),
    T2 = c(B1 = 0.7, B2 = 0.6, B3 = 0.9)
  ),
  blocks = c(B1 = 0.3, B2 = 0.4, B3 = 0.3),
  dates_of_assignment = lubridate::ymd("2023-04-15") +
    0:24 * months(1),
  time_model = time_model
) |>
  single_mab_simulation(
    assignment_method = "date",
    time_unit = "month",
    period_length = 1,
    algorithm = "Thompson",
    prior_periods = "All",
    perfect_assignment = FALSE,
    whole_experiment = FALSE,
    blocking = FALSE,
    data_cols = c(
      id_col = "id",
      success_col = "success",
      condition_col = "treatment",
      date_col = "assignment_date",
      assignment_date_col = "assignment_date",
      success_date_col = "success_date"
    )
  )
print(result)
result$estimates[1:3, ]
```

`multiple_mab_simulation` *Run Multiple Multi-Arm-Bandit Trials with Inference in Parallel*

Description

Performs multiple Multi-Arm Bandit Trials using the same simulation and inference backend as `single_mab_simulation()`. Allows for easy execution of multiple trials under the same settings to gauge the variance of the procedure across execution states. Additionally supports parallel processing through the future and furrr packages.

Usage

```
multiple_mab_simulation(
  data,
  assignment_method,
  algorithm,
  prior_periods,
  perfect_assignment,
```

```
  whole_experiment,
  blocking,
  data_cols,
  times,
  seeds,
  control_augment = 0,
  random_assign_prop = 0,
  ndraws = 5000,
  control_condition = NULL,
  time_unit = NULL,
  period_length = NULL,
  block_cols = NULL,
  verbose = FALSE,
  check_args = TRUE,
  keep_data = FALSE
)
```

Arguments

data	A data.frame, data.table, or tibble containing input data from the trial. This should be the results of a traditional Randomized Controlled Trial (RCT). Any data.frames will be converted to tibbles internally.
assignment_method	A character string; one of "date", "batch", or "individual", to define the assignment into treatment waves. When using "batch" or "individual", ensure your dataset is pre-arranged in the proper order observations should be considered so that groups are assigned correctly. For "date", observations will be considered in chronological order. "individual" assignment can be computationally intensive for larger datasets.
algorithm	A character string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1", ignoring case. Algorithm defines the adaptive assignment process. Mathematical details on these algorithms can be found in Kuleshov and Precup 2014 and Slivkins 2024.
prior_periods	A numeric value of length 1, or the string "All"; number of previous periods to use in the treatment assignment model. This is used to implement the stationary/non-stationary bandit. For example, a non-stationary bandit assumes the true probability of success for each treatment changes over time, so to account for that, not all prior data should be used when making decisions because it could be "out of date".
perfect_assignment	Logical; if TRUE, assumes perfect information for treatment assignment (i.e., all outcomes are observed regardless of the date). If FALSE, hides outcomes not yet theoretically observed, based on the dates treatments would have been assigned for each wave. This is useful when simulating batch-based assignment where treatments were assigned on a given day whether or not all the information from a prior batch was available and you have exact dates treatments were assigned.
whole_experiment	Logical; if TRUE, uses all past experimental data for imputing outcomes. If

FALSE, uses only data available up to the current period. In large datasets or with a high number of periods, setting this to FALSE can be more computationally intensive, though not a significant contributor to total run time.

blocking — Logical; whether or not to use treatment blocking. Treatment blocking is used to ensure an even-enough distribution of treatment conditions across blocks. For example, blocking by gender would mean the randomized assignment should split treatments evenly not just throughout the sample (so for 4 arms, 25-25-25-25), but also within each block, so 25% of men would receive each treatment and 25% of women the same.

data_cols — A named character vector containing the names of columns in data as strings:

- id_col: Column in data; contains unique ID as a key.
- success_col: Column in data; binary successes from the original experiment.
- condition_col: Column in data; original treatment condition for each observation.
- date_col: Column in data; contains original date of event/trial. Only necessary when assigning by "Date". Must be of type Date, not a character string.
- month_col: Column in data; contains month of treatment. Only necessary when time_unit = "Month", and when periods should be determined directly by the calendar months instead of month based time periods. This column can be a string/factor variable with the month names or numeric with the month number. It can easily be created from your date_col via lubridate::month(data[[date_col]]) or format(data[[date_col]], "%m").
- success_date_col: Column in data; contains original dates each success occurred. Only necessary when perfect_assignment = FALSE. Must be of type Date, not a character string.
- assignment_date_col: Column in data; contains original dates treatments were assigned to observations. Only necessary when perfect_assignment = FALSE. Used to simulate imperfect information on the part of researchers conducting an adaptive trial. Must be of type Date, not a character string.

times — A numeric value of length 1, the number of simulations to conduct.

seeds — An integer vector of length(times) containing valid seeds to define random state for each trial.

control_augment — A numeric value ranging from 0 to 1; proportion of each wave guaranteed to receive the "Control" treatment. Default is 0. It is not recommended to use this in conjunction with random_assign_prop.

random_assign_prop — A numeric value ranging from 0 to 1; proportion of each wave to be assigned new treatments randomly, 1 - random_assign_prop is the proportion assigned through the bandit procedure. For example if this is set to 0.1, then for each wave 10% of the observations will be randomly assigned to a new treatment, while the remaining 90% will be assigned according to UCB1 or Thompson result. It is not recommended to use this in conjunction with control_augment.

If batch sizes are small, and the number of rows is calculate to be less than 1, and probability sampling approach is used where each row in the batch will have a random_assign_prop probability of being selected for random assignment. Otherwise the number is rounded to a whole number, and that many rows are selected for random assignment.

ndraws
: A numeric value; When Thompson sampling direct calculations fail, draws from a simulated posterior will be used to approximate the Thompson sampling probabilities. This is the number of simulations to use, the default is 5000 to match the default parameter bandit::best_binomial_bandit_sim(), but might need to be raised or lowered depending on performance and accuracy concerns.

control_condition
: Value of the control condition. Only necessary when control_augment is greater than 0. Internally this value is coerced to a string, so it should be passed as a string, or a type that can easily be converted to a string.

time_unit
: A character string specifying the unit of time for assigning periods when assignment_method ="date". Acceptable values are "day", "week", or "month". "month" does not require an additional column with the months of each observation, but it can accept a separate month_col. If month_col is specified, the periods follow the calendar months strictly, and when it is not specified months are simply used as the time interval. For example if a dataset has dates starting on July 26th, under month based assignment and a specified month_col the dates July 26th and August 3st would be in different periods, but if the month_col was not specified, they would be in the same period because the dates are less than one month apart.

period_length
: A numeric value of length 1; represents the length of each treatment period. If assignment_method is "date", this length refers the number of units specified in time_unit (i.e., if "day"``, 10 would be 10 days). If assignment_method="batch"‘, this refers to the number of people in each batch.

block_cols
: A character vector of variables to block by. This vector should not be named.

verbose
: Logical; Toggles progress bar from furrr::future_map() and other intermediate messages.

check_args
: Logical; Whether or not to robustly check whether arguments are valid. Default is TRUE, and recommended not to be changed.

keep_data
: Logical; Whether or not to keep the final data from each trial. Recommended FALSE.

Details

Note that when called if data.table has not been attached already it will be when future.map() runs and a message may print. This does not mean that if you pass a tibble or data.frame, that data.table will used.

Implementation:

This function simulates multiple adaptive Multi-Arm-Bandit Trials, using experimental data from a traditional randomized experiment. It follows the same core procedure as single_mab_simulation()

(see details, there for a description), but conducts more than one simulation. This allows researchers to gauge the variance of the simulation procedure itself, and use that to form an empirical sampling distribution of the AIPW estimates, instead of relying around asymptotic normality [Hadad et al. (2021)] for inference.

The settings specified here have the same meaning as in `single_mab_simulation()`, outside of the additional parameters like `times` and `seeds` which define the number of multiple trials and random seeds to ensure reproducibility. An important note is that `seeds` can only take integer values, so they must be declared or coerced as valid integers, passing doubles (even ones that are mathematical integers) will result in an error. It is recommended to use `sample.int()`, with a known seed beforehand to generate the values. Additionally, it is highly recommended to set `keep_data = FALSE` as the memory used by the function will exponentially increase. This can cause significant performance issues, especially if your system must swap to disk because memory is full.

Parallel Processing:

The function provides support for parallel processing via the future and furrr packages. When conducting a large number of simulations, parallelization can improve performance if sufficient system resources are available. Parallel processing must be explicitly set by the user, through `future::plan()`. Windows users should set the plan to "multisession", while Linux and MacOS users can use "multicore" or "multisession". Users running in a High Performance Computing environment (HPC), are encouraged to use future.batchtools, for their respective HPC scheduler. Note that parallel processing is not guaranteed to work on all systems, and may require additional setup or debugging effort from the user. For any issues, users are encouraged to consult the documentation of the above packages.

Value

An object of class `multiple.mab`, containing:

- `final_data_nest`: A `tibble` or `data.table` containing the nested `tibbles/data.tables` from each trial. Only provided when `keep_data = TRUE`.
- `bandits`: A `tibble` or `data.table` containing the UCB1 values or Thompson sampling posterior distributions for each period. Wide format, each row is a period, and each columns is a treatment. Each row in this table represents the calculation from the given period after its values were imputed, so row 2 represents the calculations made in period 3, but represent the impact of period 2's new assignments.
- `assignment_probs`: A `tibble` or `data.table` containing the probability of being assigned each treatment arm at a given period. Wide format, each row is a period, and each columns is a treatment. Each row represents the probability of being assigned each treatment at each period, these have not been shifted like the bandits table.
- `estimates`: A `tibble` or `data.table` containing the AIPW (Augmented Inverse Probability Weighting) treatment effect estimates and variances, and traditional sample means and variances, for each treatment arm. Long format, treatment arm, and estimate type are columns along with the mean and variance.
- `assignment_quantities`: A `tibble` or `data.table` containing the number of units assigned to each treatment for each simulation in the set of repeated simulations.
- `settings`: A named list of the configuration settings used in the trial.

References

Bengtsson, Henrik. 2025. "Future: Unified Parallel and Distributed Processing in R for Everyone." https://cran.r-project.org/package=future.

Bengtsson, Henrik. 2025. "Future.Batchtools: A Future API for Parallel and Distributed Processing Using 'Batchtools.'" https://cran.r-project.org/package=future.batchtools.

Hadad, Vitor, David A. Hirshberg, Ruohan Zhan, Stefan Wager, and Susan Athey. 2021. "Confidence Intervals for Policy Evaluation in Adaptive Experiments." *Proceedings of the National Academy of Sciences of the United States of America* 118 (15): e2014602118. doi:10.1073/pnas.2014602118.

Kuleshov, Volodymyr, and Doina Precup. 2014. "Algorithms for Multi-Armed Bandit Problems." *arXiv*. doi:10.48550/arXiv.1402.6028.

Loecher, Thomas Lotze and Markus. 2022. "Bandit: Functions for Simple a/B Split Test and Multi-Armed Bandit Analysis." https://cran.r-project.org/package=bandit.

Offer-Westort, Molly, Alexander Coppock, and Donald P. Green. 2021. "Adaptive Experimental Design: Prospects and Applications in Political Science." *American Journal of Political Science* 65 (4): 826–44. doi:10.1111/ajps.12597..

Slivkins, Aleksandrs. 2024. "Introduction to Multi-Armed Bandits." *arXiv*. doi:10.48550/arXiv.1904.07272.

Vaughan, Davis, Matt Dancho, and RStudio. 2022. "Furrr: Apply Mapping Functions in Parallel Using Futures." https://cran.r-project.org/package=furrr.

See Also

single_mab_simulation(), furrr, future

Examples

```
# Multiple_mab_simulation() is a useful tool for running multiple trials
# using the same configuration settings, in different random states
data(tanf)
tanf <- tanf[1:50, ]

# The seeds passed must be integers, so it is highly recommended to create them
# before using `sample.int()`
seeds <- sample.int(10000, 5)

## Sequential Execution
x <- multiple_mab_simulation(
  data = tanf,
  assignment_method = "Batch",
  period_length = 25,
  whole_experiment = TRUE,
  blocking = FALSE,
  perfect_assignment = TRUE,
  algorithm = "Thompson",
  prior_periods = "All",
  control_augment = 0,
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
```

```
    success_col = "success"
  ),
  verbose = FALSE, times = 5, seeds = seeds, keep_data = FALSE
)
print(x)

## Parallel Execution using future:
## Check the future and furrr documentation for more details on possible options
if (requireNamespace("future", quietly = TRUE)) {
    # Set a Proper "plan"
    future::plan("multisession", workers = 2)
    multiple_mab_simulation(
      data = tanf,
      assignment_method = "Batch",
      period_length = 25,
      whole_experiment = TRUE,
      blocking = FALSE,
      perfect_assignment = TRUE,
      algorithm = "Thompson",
      prior_periods = "All",
      control_augment = 0,
      data_cols = c(
        condition_col = "condition",
        id_col = "ic_case_id",
        success_col = "success"
      ),
      verbose = FALSE, times = 5, seeds = seeds, keep_data = TRUE
    )
    # Always Set back to sequential to close processes
   future::plan("sequential")
}
```

`plot.mab` *Plot Generic for* `mab` *objects*

Description

Uses `ggplot2::ggplot()` to plot the results of a single Multi-Arm-Bandit trial. Provides options to select the type of plot, and to change how the plot looks. Objects created can be added to with + like any other ggplot plot, but arguments to change the underlying geom must be passed to the function initially.

Usage

```
## S3 method for class 'mab'
plot(x, type, level = 0.95, save = FALSE, path = NULL, ...)
```

Arguments

x	A mab class object created by single_mab_simulation()
type	String; Type of plot requested; valid types are: • arm: Shows Thompson sampling probabilities or UCB1 values over the trial period. • assign: Shows cumulative assignment proportions over the trial period. • estimate: Shows AIPW estimates for success probability with user specified normal confidence intervals based on their estimated variance.
level	Numeric value of length 1; indicates confidence interval Width (i.e 0.90, 0.95, 0.99). Defaults to 0.95.
save	Logical; Whether or not to save the plot to disk; FALSE by default.
path	String; File directory to save file if necessary.
...	Arguments to pass to ggplot2::geom_* function (e.g. color, linewidth, alpha, bins etc.).

Details

This function provides minimalist plots to quickly view the results of any Multi-Arm-Bandit trial, and has the ability to be customized through the ... inside the call and + afterwards. However, all the data necessary is provided in the output of single_mab_simulation() for extreme customization or professional plots, it is highly recommended to start completely from scratch and not use the generic.

The confidence intervals applied follow a standard normal distribution because it is assumed the AIPW estimators are asymptotically normal as shown in Hadad et al. (2021)

Value

Minimal ggplot object, that can be customized and added to with + (to change scales, labels, legend, theme, etc.).

References

Hadad, Vitor, David A. Hirshberg, Ruohan Zhan, Stefan Wager, and Susan Athey. 2021. "Confidence Intervals for Policy Evaluation in Adaptive Experiments." *Proceedings of the National Academy of Sciences of the United States of America* 118 (15): e2014602118. doi:10.1073/pnas.2014602118.

Examples

```
# Objects returned by `single_mab_simulation()` have a `mab` class.
# This class has a plot generic that has several minimal plots to examine
# the trial quickly

# Loading Data and running a quick simulation
data(tanf)
x <- single_mab_simulation(
  data = tanf,
  algorithm = "Thompson",
```

```
  assignment_method = "Batch",
  period_length = 600,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  blocking = FALSE,
  prior_periods = "All",
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success"
  )
)

# View best treatment arms over the simulation
y <- plot(x, type = "arm")
y
# Adding a new title
y + ggplot2::labs(title = "Your New Title")
# type = assign creates a similar plot, but shows probability of assignment instead

# Plotting Augmented Inverse Probability Estimates with confidence interval
# By default it provides 95% Normal Confidence Intervals but this can be adjusted
plot(x, type = "estimate")

# Adjusting height of internal geom* argument. (`geom_errorbarh()`)
plot(x, type = "estimate", height = 0.4)
```

`plot.multiple.mab` *Plot Generic For* `multiple.mab` *Objects*

Description

Uses `ggplot2::ggplot()` to plot the results of multiple Multi-Arm-Bandit trials.

Usage

```
## S3 method for class 'multiple.mab'
plot(
  x,
  type,
  quantity,
  cdf = NULL,
  level = 0.95,
  save = FALSE,
  path = NULL,
  ...
)
```

Arguments

x	A multiple.mab class object created by multiple_mab_simulation().
type	String; Type of plot requested; valid types are: • summary: Shows the number of times each arm was selected as the highest chance of being the best. • hist: Shows histograms for each treatment condition's proportion of success across trials or number of obersvations assigned. • estimate: Shows proportion of success AIPW estimates using specified normal or empirical confidence intervals.
quantity	The quantities to plot when type = "hist", accepts either 'estimate' to plot the distributuons of the AIPW estimates, or 'assignment' to plot the distributions of the number of observations assigned to each treatment across the repeated trials.
cdf	String; specifies the type of CDF to use when analyzing the estimates. valid CDFs are the 'empirical' CDF, the 'normal' CDF. Used when type = estimate. The 'normal' CDF uses the fact that the AIPW estimates are asymptotically normal, while the empirical CDF (eCDF) estimates the CDF from the sample of AIPW estimates.
level	Numeric value of length 1; indicates confidence interval Width (i.e 0.90, 0.95, 0.99). Defaults to 0.95.
save	Logical; Whether or not to save the plot to disk; FALSE by default.
path	String; File directory to save file.
...	Arguments to pass to ggplot2::geom_* function (e.g. color, linewidth, alpha, bins etc.). In the case of type = "hist", additional arguments must be passed in to distinct lists, one named geom which are passed to ggplot2::geom_* and one named facet which are passed to ggplot2::facet_grid.

Details

This function provides minimalist plots to quickly view the results of the procedure and has the ability to be customized through the ... in the call and + afterwords. However, all the data necessary is provided in the output of multiple_mab_simulation() for extreme customization or professional plots, it is highly recommended to start completely from scratch and not use the generic.

Value

Minimal ggplot object, that can be customized and added to with + (to change scales, labels, legend, theme, etc.).

Examples

```
# Objects returned by `single_mab_simulation()` have a `mab` class.
# This class has a plot generic has several minimal plots to examine the trials
# quickly
#
#
data(tanf)
```

```
tanf <- tanf[1:20, ]
# Simulating a few trials

seeds <- sample.int(100, 5)
conditions <- as.character(unique(tanf$condition))
x <- multiple_mab_simulation(
  data = tanf,
  assignment_method = "Batch",
  period_length = 10,
  whole_experiment = TRUE,
  blocking = FALSE,
  perfect_assignment = TRUE,
  algorithm = "Thompson",
  prior_periods = "All",
  control_augment = 0,
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success"
  ),
  verbose = FALSE,
  times = 5,
  seeds = seeds,
  keep_data = FALSE
)

# View number of times each treatment was the best.
plot(x, type = "summary")

# View a histogram of the AIPW estimates for each treatment.
plot(x, type = "hist", quantity = "estimate")

# Plotting AIPW confidence intervals using the empirical cdf, from the simulated
# trials.
plot(x, type = "estimate", cdf = "empirical")

# Changing the title, like any ggplot2 object.
plot(x, type = "summary") + ggplot2::labs(title = "Your New Title")

# Changing the bin width of the histograms.
plot(x, type = "hist", quantity = "assignment", geom = list(binwidth = 0.05))
```

`print.mab` *Print Generic For* `mab`

Description

Custom Print Display for objects of mab class returned by `single_mab_simulation()`. Prevents the large list from being printed directly, and provides useful information about the settings of each trial.

Usage

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

Arguments

x	A mab class object created by single_mab_simulation().
...	Further arguments passed to or from other methods.

Details

The items used to create the text summary can be found in the settings element of the output object.

... is provided to be compatible with print(), but no other arguments change the output.

Value

Text summary of settings used for the Multi-Arm Bandit trial.

Examples

```
# Running a Trial
x <- single_mab_simulation(
  data = tanf,
  algorithm = "thompson",
  assignment_method = "batch",
  period_length = 1750,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  data_cols = c(
    id_col = "ic_case_id",
    success_col = "success",
    condition_col = "condition"
  )
)
print(x)
```

---

print.multiple.mab *Print Generic For* multiple.mab

Description

Custom Print Display for multiple.mab objects returned by multiple_mab_simulation(). Prevents the large list output from being printed directly, and provides useful information about the settings for the trials.

Usage

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

Arguments

x	A multiple.mab class object created by multiple_mab_simulation().
...	Further arguments passed to or from other methods.

Details

The items used to create the text summary can be found in the settings element of the output object.

... is provided to be compatible with print(), no other arguments change output.

Value

Text summary of settings used for the Multi-Arm Bandit trials.

Examples

```
# Running Multiple Simulations
x <- multiple_mab_simulation(
  data = tanf,
  algorithm = "thompson",
  assignment_method = "Batch",
  period_length = 1750,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  data_cols = c(
    id_col = "ic_case_id",
    success_col = "success",
    condition_col = "condition"
  ),
  times = 5, seeds = sample.int(5)
)
print(x)
```

---

single_mab_simulation *Run One Adaptive Simulation With Inference.*

Description

Performs a single Multi-Arm Bandit (MAB) trial using experimental data from an original randomized controlled trial, and adaptive inference strategies as described in Hadad et al. (2021). Wraps around the internal implementation functions, and performs the full MAB pipeline: preparing inputs, assigning treatments and imputing successes, and adaptively weighted estimation. See the details and vignettes to learn more.

Usage

```
single_mab_simulation(
  data,
  assignment_method,
  algorithm,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  control_augment = 0,
  random_assign_prop = 0,
  ndraws = 5000,
  control_condition = NULL,
  time_unit = NULL,
  period_length = NULL,
  block_cols = NULL,
  verbose = FALSE,
  check_args = TRUE
)
```

Arguments

`data`	A `data.frame`, `data.table`, or `tibble` containing input data from the trial. This should be the results of a traditional Randomized Controlled Trial (RCT). Any `data.frames` will be converted to tibbles internally.
`assignment_method`	A character string; one of `"date"`, `"batch"`, or `"individual"`, to define the assignment into treatment waves. When using `"batch"` or `"individual"`, ensure your dataset is pre-arranged in the proper order observations should be considered so that groups are assigned correctly. For "date", observations will be considered in chronological order. `"individual"` assignment can be computationally intensive for larger datasets.
`algorithm`	A character string specifying the MAB algorithm to use. Options are `"Thompson"` or `"UCB1"`, ignoring case. Algorithm defines the adaptive assignment process. Mathematical details on these algorithms can be found in Kuleshov and Precup 2014 and Slivkins 2024.
`prior_periods`	A numeric value of length 1, or the string `"All"`; number of previous periods to use in the treatment assignment model. This is used to implement the stationary/non-stationary bandit. For example, a non-stationary bandit assumes the true probability of success for each treatment changes over time, so to account for that, not all prior data should be used when making decisions because it could be "out of date".
`perfect_assignment`	Logical; if `TRUE`, assumes perfect information for treatment assignment (i.e., all outcomes are observed regardless of the date). If `FALSE`, hides outcomes not yet theoretically observed, based on the dates treatments would have been assigned

for each wave. This is useful when simulating batch-based assignment where treatments were assigned on a given day whether or not all the information from a prior batch was available and you have exact dates treatments were assigned.

`whole_experiment`

Logical; if `TRUE`, uses all past experimental data for imputing outcomes. If `FALSE`, uses only data available up to the current period. In large datasets or with a high number of periods, setting this to `FALSE` can be more computationally intensive, though not a significant contributor to total run time.

`blocking` Logical; whether or not to use treatment blocking. Treatment blocking is used to ensure an even-enough distribution of treatment conditions across blocks. For example, blocking by gender would mean the randomized assignment should split treatments evenly not just throughout the sample (so for 4 arms, 25-25-25-25), but also within each block, so 25% of men would receive each treatment and 25% of women the same.

`data_cols` A named character vector containing the names of columns in `data` as strings:

- `id_col`: Column in `data`; contains unique ID as a key.
- `success_col`: Column in `data`; binary successes from the original experiment.
- `condition_col`: Column in `data`; original treatment condition for each observation.
- `date_col`: Column in `data`; contains original date of event/trial. Only necessary when assigning by "Date". Must be of type `Date`, not a character string.
- `month_col`: Column in `data`; contains month of treatment. Only necessary when `time_unit = "Month"`, and when periods should be determined directly by the calendar months instead of month based time periods. This column can be a string/factor variable with the month names or numeric with the month number. It can easily be created from your `date_col` via `lubridate::month(data[[date_col]])` or `format(data[[date_col]], "%m")`.
- `success_date_col`: Column in `data`; contains original dates each success occurred. Only necessary when `perfect_assignment = FALSE`. Must be of type `Date`, not a character string.
- `assignment_date_col`: Column in `data`; contains original dates treatments were assigned to observations. Only necessary when `perfect_assignment = FALSE`. Used to simulate imperfect information on the part of researchers conducting an adaptive trial. Must be of type `Date`, not a character string.

`control_augment`

A numeric value ranging from 0 to 1; proportion of each wave guaranteed to receive the "Control" treatment. Default is 0. It is not recommended to use this in conjunction with `random_assign_prop`.

`random_assign_prop`

A numeric value ranging from 0 to 1; proportion of each wave to be assigned new treatments randomly, 1 - `random_assign_prop` is the proportion assigned through the bandit procedure. For example if this is set to 0.1, then for each wave 10% of the observations will be randomly assigned to a new treatment,

while the remaining 90% will be assigned according to UCB1 or Thompson result. It is not recommended to use this in conjunction with `control_augment`. If batch sizes are small, and the number of rows is calculate to be less than 1, and probability sampling approach is used where each row in the batch will have a `random_assign_prop` probability of being selected for random assignment. Otherwise the number is rounded to a whole number, and that many rows are selected for random assignment.

`ndraws` A numeric value; When Thompson sampling direct calculations fail, draws from a simulated posterior will be used to approximate the Thompson sampling probabilities. This is the number of simulations to use, the default is 5000 to match the default parameter `bandit::best_binomial_bandit_sim()`, but might need to be raised or lowered depending on performance and accuracy concerns.

`control_condition`
Value of the control condition. Only necessary when `control_augment` is greater than 0. Internally this value is coerced to a string, so it should be passed as a string, or a type that can easily be converted to a string.

`time_unit` A character string specifying the unit of time for assigning periods when `assignment_method ="date"`. Acceptable values are `"day"`, `"week"`, or `"month"`. `"month"` does not require an additional column with the months of each observation, but it can accept a separate `month_col`. If `month_col` is specified, the periods follow the calendar months strictly, and when it is not specified months are simply used as the time interval. For example if a dataset has dates starting on July 26th, under month based assignment and a specified `month_col` the dates July 26th and August 3st would be in different periods, but if the `month_col` was not specified, they would be in the same period because the dates are less than one month apart.

`period_length` A numeric value of length 1; represents the length of each treatment period. If `assignment_method` is "date", this length refers the number of units specified in `time_unit` (i.e., if `"day"``, 10 would be 10 days). If assignment_method="batch"‘, this refers to the number of people in each batch.

`block_cols` A character vector of variables to block by. This vector should not be named.

`verbose` Logical; whether or not to print intermediate messages. Default is `FALSE`.

`check_args` Logical; Whether or not to robustly check whether arguments are valid. Default is TRUE, and recommended not to be changed.

Details

For all the items laballed as a `tibble` or `data.table`, `data.tables` will be used if the user passed `data` is a `data.table`, `tibbles` used otherwise.

Implementation:

At each period, either the Thompson sampling probabilities or UCB1 values are calculated based on the outcomes from the number of `prior_periods` specified. New treatments are then assigned randomly using the Thompson sampling probabilities via the randomizr package, or as the treatment with the highest UCB1 values, while implementing the specific treatment blocking and

control augmentation specified. More details on bandit algorithms can in Kuleshov and Precup 2014 and Slivkins 2024.

If a hybrid assignment is specified, here is where it is implemented in the simulation. `control_augment` is a threshold probability for the control group, and the assignment probabilities are changed to ensure that threshold is met. The other hybrid assignment is `random_assign_prop`. Here, the specified proportion of the data is set aside to assign treatments randomly, while the rest of the data is assigned through the bandit procedure.

After assigning treatments, observations with new treatments have their outcomes imputed, with any specified treatment blocking. The probabilities of success used to impute, are estimated via the grouped means of successes from the original data either from the whole trial, or up to that period, defined by `whole_experiment`.

If `perfect_assignment = FALSE`, new dates of success will be imputed using averages of those dates in the period, grouped by treatment block. Observations for which their treatment changed, but their outcome was success in the original and simulation, do not have their date changed. When the next period starts, the success dates are checked against the maximum/latest `assignment_date` for the period, and if any success occurs after that, it is treated as a failure for the purpose of the bandit decision algorithms.

At the end of the simulation the results are aggregated together to calculate the Adaptively Weighted Augmented Inverse Probability Estimator (Hadad et al. 2021) using the mean and variance formulas provided, under the constant allocation rate adaptive schema. These estimators are unbiased and asymptotically normal under the adaptive conditions which is why they are used. For a complete view of their properties, reading the paper is recommended.

Performance Concerns:

This procedure has the potential to be computationally expensive and time-consuming. Performance depends on the relative size of each period, number of periods, and overall size of the dataset. This function has separate support for `data.frames` and `data.tables`. If a `data.frame` is passed, the function uses a combination of `dplyr`, `tidyr` and base `R` to shape data, and run the simulation. However, if a `data.table` is passed the function exclusively uses the `data.table` code for all the same operations.

In general, smaller batches run faster under base `R`, while larger ones could benefit from the performance and memory efficiencies provided by `data.table`. However, we've observed larger datasets can cause numerical instability with some calculations in the Thompson sampling procedure. Internal safeguards exist to prevent this, but the best way to preempt any issues is to set `prior_periods` to a low number.

For more information about how to use the function, please view the vignette.

Value

An object of class `mab`, containing:

- `final_data`: The processed `tibble` or `data.table`, containing new columns pertaining to the results of the trial. Specifically Contains:
  - `period_number`: Assigned period for simulation.
  - `mab_*`: New treatment conditions and outcomes under the simulation.
  - `impute_req`: Whether observation required an imputed outcome.
  - `*block`: variables relating to the block specified for treatment blocking, and the concatenation of that block with an observations original treatment, and new treatment.

  - `aipw_*` Columns containing individual Augmented Inverse Probability Weighted estimates for each observation and treatment arm.
  - `prior_rate_*`: Columns containing success rate for each treatment arm, from all periods before the observations period of the simulation.
  - `*_assign_prob`: Columns containing probability of being assigned each treatment at the given period.

- `bandits`: A `tibble` or `data.table` containing the UCB1 values or Thompson sampling posterior distributions for each period. Wide format, each row is a period, and each columns is a treatment. Each row in this table represents the calculation from the given period after its values were imputed, so row 2 represents the calculations made in period 3, but represent the impact of period 2's new assignments.
- `assignment_probs`: A `tibble` or `data.table` containing the probability of being assigned each treatment arm at a given period. Wide format, each row is a period, and each columns is a treatment. Each row represents the probability of being assigned each treatment at each period, these have not been shifted like the bandits table.
- `estimates`: A `tibble` or `data.table` containing the AIPW (Augmented Inverse Probability Weighting) treatment effect estimates and variances, and traditional sample means and variances, for each treatment arm. Long format, treatment arm, and estimate type are columns along with the mean and variance.
- `settings`: A named list of the configuration settings used in the trial.

References

Hadad, Vitor, David A. Hirshberg, Ruohan Zhan, Stefan Wager, and Susan Athey. 2021. "Confidence Intervals for Policy Evaluation in Adaptive Experiments." *Proceedings of the National Academy of Sciences of the United States of America* 118 (15): e2014602118. doi:10.1073/pnas.2014602118.

Kuleshov, Volodymyr, and Doina Precup. 2014. "Algorithms for Multi-Armed Bandit Problems." *arXiv*. doi:10.48550/arXiv.1402.6028.

Loecher, Thomas Lotze and Markus. 2022. "Bandit: Functions for Simple a/B Split Test and Multi-Armed Bandit Analysis." https://cran.r-project.org/package=bandit.

Offer-Westort, Molly, Alexander Coppock, and Donald P. Green. 2021. "Adaptive Experimental Design: Prospects and Applications in Political Science." *American Journal of Political Science* 65 (4): 826-44. doi:10.1111/ajps.12597.

Slivkins, Aleksandrs. 2024. "Introduction to Multi-Armed Bandits." *arXiv*. doi:10.48550/arXiv.1904.07272.

See Also

`multiple_mab_simulation()`, `summary.mab()`, `plot.mab()`.

Examples

```
# Loading Example Data and defining conditions
data(tanf)

## Running Thompson sampling with 500 person large batches,
## with no blocks and imperfect assignment
```

```
single_mab_simulation(
  data = tanf,
  assignment_method = "Batch",
  algorithm = "Thompson",
  period_length = 500,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = FALSE,
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success",
    success_date_col = "date_of_recert",
    assignment_date_col = "letter_sent_date"
  )
)

## Running UCB1 Sampling with 1 Month based batches and
## control augmentation set to 0.25, with perfect_assignment.
## When using control_augment > 0, conditions need to have proper names
# no_letter is control, the others are treatments

single_mab_simulation(
  data = tanf,
  assignment_method = "Date",
  time_unit = "Month",
  algorithm = "UCB1",
  period_length = 1,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  control_condition = "no_letter",
  control_augment = 0.25,
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success",
    date_col = "appt_date",
    month_col = "recert_month"
  )
)

## 5 Day Periods with Thompson, Treatment Blocking by Service Center,
## Whole experiment FALSE, and hybrid assignment 10% random, 90% bandit.
single_mab_simulation(
  data = tanf,
  assignment_method = "Date",
  time_unit = "Day",
  algorithm = "Thompson",
  period_length = 5,
  prior_periods = "All",
```

```
  blocking = TRUE,
  block_cols = c("service_center"),
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  random_assign_prop = 0.1,
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success",
    date_col = "appt_date"
  )
)
```

---

summary.mab *Summary Generic For* mab *Class*

Description

Summarizes the Results of a Single Multi-Arm Bandit Trial. Provides confidence intervals around the AIPW estimates, final calculations of the Thompson sampling probabilities or UCB1 values, and the number of observations assigned for each arm.

Usage

```
## S3 method for class 'mab'
summary(object, level = 0.95, ...)
```

Arguments

object	A mab class object created by single_mab_simulation().
level	Numeric value of length 1; indicates confidence interval Width (i.e 0.90, 0.95, 0.99). Defaults to 0.95.
...	Additional arguments.

Details

The confidence intervals applied follow a standard normal distribution because it is assumed the AIPW estimators are asymptotically normal as shown in Hadad et al. (2021).

... is provided to be compatible with summary(), the function does not have any additional arguments.

All of the data provided to create a table like this is present in the object created by single_mab_simulation() but this provides a simple shortcut, which is useful when testing many different simulations.

Value

A tibble containing summary information from the trial with the columns:

- `Treatment_Arm`: Contains the treatment condition.
- `Probability_Of_Best_Arm/UCB1_Value`: Final Thompson sampling probabilities or UCB1 values for each treatment.
- `estimated_probability_of_success`: The AIPW estimates for the probability of success for each treatment.
- `SE`: The standard error for the AIPW estimates.
- `lower_bound`: The lower bound on the normal confidence interval for the `estimated_probability_of_success`. Default is 95%.
- `upper_bound`: The upper bound on the normal confidence interval for the `estimated_probability_of_success`. Default is 95%.
- `num_assigned`: The number of observations assigned to each treatment under the simulated trial.
- `level`: The confidence level for the confidence interval, default is 95%.
- `periods`: The total number of periods of the simulation.

References

Hadad, Vitor, David A. Hirshberg, Ruohan Zhan, Stefan Wager, and Susan Athey. 2021. "Confidence Intervals for Policy Evaluation in Adaptive Experiments." *Proceedings of the National Academy of Sciences of the United States of America* 118 (15): e2014602118. doi:10.1073/pnas.2014602118.

Examples

```
# Objects returned by `single_mab_simulation()` have a `mab` class.
# This class has a summary generic that can produce quick results of the trial.

# Loading Data and running a quick simulation
data(tanf)
x <- single_mab_simulation(
  data = tanf,
  algorithm = "Thompson",
  assignment_method = "Batch",
  period_length = 600,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  blocking = FALSE,
  prior_periods = "All",
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success"
  )
)

# Creating summary table
```

```
## Defaults to 95% confidence interval
summary(x) |> print(width = Inf)

## 70% confidence level
summary(x, level = 0.7) |> print(width = Inf)
```

`summary.multiple.mab` *Summary Generic For* `multiple.mab` *Class*

Description

Summarizes results of multiple Multi-Arm Bandit Trials. Provides empirically estimated and normally approximated confidence intervals on AIPW estimates for probability of success, the number of times each arm was the chosen as the best treatment across all simulations, and the average for how many units were assigned to each treatment across all the simulations.

Usage

```
## S3 method for class 'multiple.mab'
summary(object, level = 0.95, ...)
```

Arguments

`object`	A `multiple.mab` object created by multiple_mab_simulation.
`level`	Numeric value of length 1; indicates confidence interval Width (i.e 0.90, 0.95, 0.99). Defaults to 0.95.
`...`	Additional arguments.

Details

The empirically estimated variances and confidence intervals, use the variance measured directly in the AIPW estimates for each treatment over all the simulations. The normal confidence intervals are estimated using an average of the measured variances across the simulations.

The best arm at the end of each trial is chosen by the highest UCB1 value or Thompson sampling probability. These values indicate which treatment would be chosen next, or have the highest probability of being chosen next, therefore representing the current best treatment.

Additionally, an average and standard deviation for the number of units assigned to each treatment across all the simulations is provided.

`...` is provided to be compatible with `summary()`, the function does not have any additional arguments.

Value

A tibble containing summary information from the repeated trials with the columns:

- `Treatment_Arm`: Contains the treatment condition.
- `average_probability_of_success`: The average of the AIPW estimates for the probability of success for each treatment across the trials.
- `SE_avg`: The standard error for the AIPW estimates, calculated as the square root of the average of the variances.
- `SE_empirical`: The standard error estimated empirically as the standard deviation of the all the calculated AIPW estimates for probability of success.
- `lower_normal`: The lower bound on the normal confidence interval for the `estimated_probability_of_success`. Default is 95%.
- `upper_normal`: The upper bound on the normal confidence interval for the `estimated_probability_of_success`. Default is 95%.
- `lower_empirical`: The lower bound on the empirical confidence interval for the `estimated_probability_of_succes` Calculated using the observed distribution of AIPW estimated probabilities of success. Default is 95%.
- `upper_empirical`: The upper bound on the empirical confidence interval for the `estimated_probability_of_succes` Calculated using the observed distribution of AIPW estimated probabilities of success. Default is 95%.
- `times_best`: The number of times each treatment arm was selected as the best for an individual simulation.
- `average_num_assigned`: The average number of observations assigned to each treatment under the simulated trials.
- `sd_num_assigned`: The standard deviation for the number of observations assigned to each treatment under the simulated trials.
- `level`: The confidence level for the confidence interval, default is 95%.

Examples

```
# Objects returned by `multiple_mab_simulation()` have a `multiple.mab` class.
# This class has a summary generic that can produce quick results of the trials
data(tanf)
tanf <- tanf[1:100, ]
# Simulating a few trials
seeds <- sample.int(10000, 5)
x <- multiple_mab_simulation(
  data = tanf,
  assignment_method = "Batch",
  period_length = 20,
  whole_experiment = TRUE,
  blocking = FALSE,
  perfect_assignment = TRUE,
  algorithm = "Thompson",
  prior_periods = "All",
  control_augment = 0,
```

```
  data_cols = c(
    condition_col = "condition",
    id_col = "ic_case_id",
    success_col = "success"
  ),
  verbose = FALSE,
  times = 5,
  seeds = seeds,
  keep_data = FALSE
)

# Creating summary table
## Defaults to 95% confidence interval
summary(x) |> print(width = Inf)

## 70% confidence level
summary(x, level = 0.7) |> print(width = Inf)
```

`tanf` *Public TANF Recipient Data From Washington D.C*

Description

A modified version of the data set used in https://thelabprojects.dc.gov/benefits-reminder-letter with one additional column added for analysis.

Usage

```
data(tanf)
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 3517 rows and 21 columns.

Details

Variables are as follows:

ic_case_id Unique, anonymized case identifier.

service_center DC Department of Human Services Center assigned each case.

condition The assigned letter condition: "No Letter", "Open Appointment", or "Specific Appointment".

recert_month Recertification Month.

letter_sent_date Date the second (treatment) letter was sent.

recert_id Administrative recertification identifier.

return_to_sender Indicates whether letter was returned as undeliverable

pdc_status PDC Status

renewal_date Date by which renewal must be completed.

notice_date.x Date the first notice was sent (initial legal communication)

days_betwn_notice_and_recert_due Number of days between the first notice and the recertification due date.

cert_period_start Start date of the recertification period.

cert_period_end End date of recertification period.

recert_status Status of recertification process (Pending, Denied, etc.)

denial_reason Reason for denial if recertification was not approved.

recert_month_year Combined recertification month and year.

notice_date.y Alternate record of first notice date.

recert_status_dcas Official recertification status from DCAS

date_of_recert Date the recertification was successfully submitted (if applicable).

success Binary variable indicating successful recertification based on recert_status (newly added column).

Source

https://github.com/thelabdc/DHS-TANFRecertification-Public/blob/main/data/df_replication_anonymized.csv

Index