Package 'whatifbandit'

July 12, 2025

Title Analyzing Randomized Experiments as Multi-Arm Bandits

Version 0.0.0.9000

Description Simulates the results of completed randomized controlled trials, as if they had been conducted as adaptive Multi-Arm Bandit (MAB) trials instead using data from the original experiment. Utilizes augmented inverse probability weighted estimatation (AIPW) to robustly estimate the probability of success for each treatment arm. Provides cutomization options to simulate perfect/imperfect information, stationary/non-stationary bandits, treatment blocking, and control augmentation strategy for assigning treatment arms.

License GPL (>= 3) + file LICENSE

Depends R (>= 4.1.0)

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

Suggests future,
ggplot2,
testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Contents

`adaptive_aipw` *Calculate Adaptive AIPW Estimates*

Description

Takes the average of the individual AIPW scores created by `get_iaipw()` for each period, and assigns each estimate an adaptive weight based on a constant allocation rate across periods defined by Hadad et. al (2021) to calculate a final estimate for each treatment condition.

Usage

```
adaptive_aipw(data, assignment_probs, conditions, periods, verbose)
```

Arguments

`data`	`final_data` object from `run_mab_trial()`. Contains results of Multi-Arm-Bandit Simulation
`assignment_probs`	`assignment_probs` object from `run_mab_trial()`. Contains probability of receiving each treatment at each treatment period in the simulated trial.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`periods`	Numeric; number of treatment waves.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.

See Also

- `get_iaipw()`
- `get_adaptive_aipw()`
- `mab_simulation()`

`adaptive_aipw.data.frame`

Adaptive AIPW Estimates for data.frames

Description

Adaptive AIPW Estimates for data.frames

Usage

```
## S3 method for class 'data.frame'
adaptive_aipw(data, assignment_probs, conditions, periods, verbose)
```

Arguments

`data`	`final_data` object from `run_mab_trial()`. Contains results of Multi-Arm-Bandit Simulation
`assignment_probs`	`assignment_probs` object from `run_mab_trial()`. Contains probability of receiving each treatment at each treatment period in the simulated trial.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`periods`	Numeric; number of treatment waves.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.

`adaptive_aipw.data.table`

Adaptive AIPW Estimates for data.tables

Description

Adaptive AIPW Estimates for data.tables

Usage

```
## S3 method for class 'data.table'
adaptive_aipw(data, assignment_probs, conditions, periods, verbose)
```

Arguments

`data`	`final_data` object from `run_mab_trial()`. Contains results of Multi-Arm-Bandit Simulation
`assignment_probs`	`assignment_probs` object from `run_mab_trial()`. Contains probability of receiving each treatment at each treatment period in the simulated trial.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`periods`	Numeric; number of treatment waves.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.

`assign_treatments` *Adaptively Assign Treatments in a Period*

Description

Assigns new treatments for an assignment wave based on the assignment probabilities provided. Probabilities passed to `randomizr::block_and_cluster_ra()` or `randomizr::cluster_ra()` for random assignment.

Usage

```
assign_treatments(
  current_data,
  probs,
  blocking = NULL,
  algorithm,
  id_col,
  conditions,
  condition_col,
  success_col
)
```

Arguments

`current_data`	Data with only observations from the current sampling period.
`probs`	Named Numeric Vector; Probability of Assignment for each treatment condition.
`blocking`	Logical; Whether or not to use treatment blocking.
`algorithm`	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
`id_col`	Column in data, contains unique id as a key.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`condition_col`	Column in data; Original Treatment condition for each observation.
`success_col`	Column in data; Binary successes from original experiment.

Value

Updated `data` object with the new treatment conditions. If this treatment is different then from under the original experiment, they are labelled as imputation required.

See Also

- `run_mab_trial()`
- `randomizr::block_and_cluster_ra()`
- `randomizr::cluster_ra()`

augment_prob *Control Augmentation for Treatment Assignment*

Description

Adjusts Probabilities of Assignment to match a control augmentation framework. If probability threshold is not meant, values are adjusted uniformly to augment the control probability.

Usage

```
augment_prob(assignment_probs, control_augment, conditions, algorithm)
```

Arguments

assignment_probs	Named numeric vector; contains probabilities of assignment with the control condition named "Control".
control_augment	Number $\in$ [0,1]; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
algorithm	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".

Value

Named numeric vector with updated probabilities

augment_prob.Thompson *Augment Prob For Thompson Sampling*

Description

Augment Prob For Thompson Sampling

Usage

```
## S3 method for class 'Thompson'
augment_prob(assignment_probs, control_augment, conditions)
```

Arguments

assignment_probs	Named numeric vector; contains probabilities of assignment with the control condition named "Control".
control_augment	Number $\in$ [0,1]; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

augment_prob.UCB1 *Augment Prob For UCB1*

Description

Augment Prob For UCB1

Usage

```
## S3 method for class 'UCB1'
augment_prob(assignment_probs, control_augment, conditions)
```

Arguments

assignment_probs	Named numeric vector; contains probabilities of assignment with the control condition named "Control".
control_augment	Number $\in$ [0,1]; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

check_args *Validates Inputs For* single_mab_simulation() *and* multiple_mab_simulation()

Description

This function provides input validation, checking to ensure that all required function arguments have been entered, and that they do not conflict with one another. The goal is to provide the user with informative error messages so they can quickly fix their usage of the function.

Usage

```
check_args(
  data,
  assignment_method,
  algorithm,
  conditions,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  block_cols,
  time_unit,
  period_length,
  control_augment,
  verbose
)
```

Arguments

data	A data frame, tibble or data.table that provides the input data for the trial.
assignment_method	String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.
algorithm	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
prior_periods	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
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.
whole_experiment	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
blocking	Logical; Whether or not to use treatment blocking.
data_cols	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 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'. • month_col: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • success_date_col: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • assignment_date_col: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.
block_cols	Character Vector of variables to block by.
time_unit	A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".
period_length	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified time_unit (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.
control_augment	Number $\in [0,1]$; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.
verbose	Logical; Whether or not to print iteration number. FALSE by default.

Value

No return value. Throws an error if problems exist before running single_mab_simulation or multiple_mab_simulation().

See Also

*single_mab_simulation() *multiple_mab_simulation()

---

check_cols *Checking existence and declaration of columns*

Description

Takes the user's settings as input, and checks the required columns against which ones are provided, and throws in error if the user did not provide a required column, or the column they provide is not present in their data.

Usage

```
check_cols(
  assignment_method,
  time_unit,
  perfect_assignment,
  data_cols,
  data,
  verbose
)
```

Arguments

`assignment_method`
: String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.

`time_unit`
: A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".

`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.

`data_cols`
: 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 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'.
- `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.

- `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.

Value

Throws an error if columns are not properly declared or in data

See Also

*`single_mab_simulation()` *`check_args()`

`check_estimator` *Check Estimator*

Description

Shorthand for checking if the `estimator` passed to plot.mab and plot.multiple.mab are valid

Usage

```
check_estimator(estimator)
```

Arguments

`estimator`	Estimator to plot; Either "AIPW", "Sample" or "Both"; only used by "estimate" type

Value

Throws an error if the argument is invalid; returns character vector with the user's selection based on the argument

check_impute *Checking Imputation Info*

Description

Ensures the Imputation Info in the current iteration of `run_mab_trial()`, contains all the info needed, important when blocking or using small assignment waves

Usage

```
check_impute(imputation_information, current_data, current_period)
```

Arguments

`imputation_information`	`success` element of the `imputation_information` list created by `imputation_prep()`.
`current_data`	Data with only observations from the current sampling period.
`current_period`	Numeric scalar; current treatment wave of the simulation.

check_impute.data.frame *`check_impute()` for data.frames*

Description

`check_impute()` for data.frames

Usage

```
## S3 method for class 'data.frame'
check_impute(imputation_information, current_data, current_period)
```

Arguments

`imputation_information`	`success` element of the `imputation_information` list created by `imputation_prep()`.
`current_data`	Data with only observations from the current sampling period.
`current_period`	Numeric scalar; current treatment wave of the simulation.

check_impute.data.table *check_impute() for data.tables*

Description

check_impute() for data.tables

Usage

```
## S3 method for class 'data.table'
check_impute(imputation_information, current_data, current_period)
```

Arguments

imputation_information	success element of the imputation_information list created by imputation_prep().
current_data	Data with only observations from the current sampling period.
current_period	Numeric scalar; current treatment wave of the simulation.

check_level *Check Level*

Description

Shorthand for Checking if the level argument in the S3 generic methods is valid for a confidence interval.

Usage

```
check_level(level)
```

Arguments

level	Confidence Interval Width (i.e 0.90, .95, 0.99)

Value

Throws an error if level is invalid, else does nothing

cols *Column arguments shared across functions*

Description

Column arguments shared across functions

Arguments

id_col	Column in data, contains unique id as a key.
success_col	Column in data; Binary successes from 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'.
month_col	Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
success_date_col	Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
assignment_date_col	Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE.

condense_results *Condenses results into a list for* multiple_mab_simulation()

Description

Takes the output from furrr::future_map() in multiple_mab_simulation() and condenses it to return to the user

Takes the output from furrr::future_map() in multiple_mab_simulation() and condenses it to return to the user

Usage

```
condense_results(data, keep_data, mabs, times)

condense_results(data, keep_data, mabs, times)
```

Arguments

data	A data frame, tibble or data.table that provides the input data for the trial.
keep_data	Logical; Whether or not to keep the final data from each trial. Recommended FALSE for large datasets .
mabs	output from furrr::future_map() in multiple_mab_simulation()
times	Integer; number of simulations to conduct.

Value

`multiple.mab` class object, which is a named list containing:

- `final_data_nest`: Data.frame containing a nested data.frame with the final data from each trial
- `bandits`: Data.frame containing the Thompson/UCB1 statistics across all treatments, periods, and trials
- `estimates`: Data.frame containing the AIPW statistics across all treatments, and trials
- `settings`: A list of the configuration settings used in the trial.

`multiple.mab` class object, which is a named list containing:

- `final_data_nest`: Data.frame containing a nested data.frame with the final data from each trial
- `bandits`: Data.frame containing the Thompson/UCB1 statistics across all treatments, periods, and trials
- `estimates`: Data.frame containing the AIPW statistics across all treatments, and trials
- `settings`: A list of the configuration settings used in the trial.

`create_cutoff` *Create Treatment Wave Cutoffs*

Description

Used during `pre_mab_simulation()` to assign each observation a new treatment assignment period, based on user-supplied specifications, and user supplied data from `date_col` and `month_col` in `data_cols`.

Usage

```
create_cutoff(
  data,
  data_cols,
  period_length = NULL,
  assignment_method,
  time_unit
)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.

- success_date_col: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- assignment_date_col: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

period_length Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified time_unit (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

assignment_method String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.

time_unit A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".

Value

Updated data object with the new period_number column. period_number is an integer representing an observation's new assignment period.

See Also

*pre_mab_simulation()

---

create_cutoff.Batch create_cutoff() *Batch Based Periods*

Description

create_cutoff() Batch Based Periods

Usage

```
## S3 method for class 'Batch'
create_cutoff(data, period_length)
```

Arguments

data A data frame, tibble or data.table that provides the input data for the trial.

period_length Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified time_unit (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

create_cutoff.Day `create_cutoff()` *Day Based Periods*

Description

`create_cutoff()` Day Based Periods

Usage

```
## S3 method for class 'Day'
create_cutoff(data, date_col, period_length)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`date_col`	Column in data, contains original date of event/trial; only necessary when assigning by 'Date'.
`period_length`	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

create_cutoff.Individual `create_cutoff()` *Individual Periods*

Description

`create_cutoff()` Individual Periods

Usage

```
## S3 method for class 'Individual'
create_cutoff(data)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.

create_cutoff.Month `create_cutoff()` *Month Based Periods*

Description

#' @method create_cutoff Month

Usage

```
create_cutoff.Month(data, date_col, month_col, period_length)
```

Arguments

data	A data frame, tibble or data.table that provides the input data for the trial.
date_col	Column in data, contains original date of event/trial; only necessary when assigning by 'Date'.
month_col	Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
period_length	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

create_cutoff.Week `create_cutoff()` *Week Based Periods*

Description

`create_cutoff()` Week Based Periods

Usage

```
## S3 method for class 'Week'
create_cutoff(data, date_col, period_length)
```

Arguments

data	A data frame, tibble or data.table that provides the input data for the trial.
date_col	Column in data, contains original date of event/trial; only necessary when assigning by 'Date'.
period_length	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

`create_new_cols` *Create Necessary Columns for Multi-Arm Bandit Trial*

Description

Initializes partially empty columns in `data`, to ensure compatibility with, `mab_simulation()`. These are initialized as `NA` except for observations with `period_number` = 1, whose are the starting point for the adaptive trial.

Usage

```
create_new_cols(data, data_cols, block_cols, blocking, perfect_assignment)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.
`block_cols`	Character Vector of variables to block by.
`blocking`	Logical; Whether or not to use treatment blocking.
`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.

Value

Updated `data` object of same class with 6 new columns:

- mab_success: New variable to hold new success from Multi-arm bandit procedure, NA until assigned.
- mab_condition: New variable to hold new treatment condition from Multi-arm bandit procedure, NA until assigned.
- impute_req: Binary indicator for imputation requirement, NA until assigned.

- new_success_date: New variable to new recertification date from Multi-arm bandit procedure, NA until assigned.
- block New variable indicating the variables to block by for assignment.
- treatment_block New variable combining block with original treatment condition.

See Also

*create_cutoff() *pre_mab_simulation()

`create_new_cols.data.frame` `create_new_cols()` *for data.frames and tibbles*

Description

`create_new_cols()` for data.frames and tibbles

Usage

```
## S3 method for class 'data.frame'
create_new_cols(data, data_cols, block_cols, blocking, perfect_assignment)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.
`block_cols`	Character Vector of variables to block by.
`blocking`	Logical; Whether or not to use treatment blocking.
`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.

`create_new_cols.data.table` *create_new_cols() for Data.tables*

Description

`create_new_cols()` for Data.tables

Usage

```
## S3 method for class 'data.table'
create_new_cols(data, data_cols, blocking, block_cols, perfect_assignment)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.
`blocking`	Logical; Whether or not to use treatment blocking.
`block_cols`	Character Vector of variables to block by.
`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.

create_prior *Create Prior Periods*

Description

Used during run_mab_trial() to create a vector of prior periods dynamically.

Usage

```
create_prior(prior_periods, current_period)
```

Arguments

prior_periods	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
current_period	The current period of the simulation. Defined by loop structure inside run_mab_trial().

Value

Numeric vector containing the prior treatment periods to be used for UCB1 and Thompson algorithms to assign treatments

See Also

*run_mab_trial()

end_mab_trial *Ends Multi-Arm Bandit Trial*

Description

Condenses output from run_mab_trial() into manageable structure.

Usage

```
end_mab_trial(data, bandits, algorithm, periods, conditions)
```

Arguments

data	finalized data from run_mab_trial().
bandits	Finalized bandits list from run_mab_trial().
algorithm	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
periods	Numeric scalar; total number of periods in Multi-Arm-Bandit trial.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

Value

A named list containing:

- `final_data`: Processed data with new treatment assignments and imputed outcomes labelled with "mab_" prefix.
- `bandits`: Thompson Probability or UCB1 statistic for each treatment arm at each period of the simulation.
- `assignment_probs`: Assignment probabilities for each treatment arm at each period of the simulation.

See Also

- `run_mab_trial()`

`end_mab_trial.data.frame`
`end_mab_trial()` *for data.frames*

Description

`end_mab_trial()` for data.frames

Usage

```
## S3 method for class 'data.frame'
end_mab_trial(data, bandits, algorithm, periods, conditions)
```

Arguments

`data`	finalized data from `run_mab_trial()`.
`bandits`	Finalized bandits list from `run_mab_trial()`.
`algorithm`	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
`periods`	Numeric scalar; total number of periods in Multi-Arm-Bandit trial.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

end_mab_trial.data.table *end_mab_trial() for data.tables*

Description

end_mab_trial() for data.tables

Usage

```
## S3 method for class 'data.table'
end_mab_trial(data, bandits, algorithm, periods, conditions)
```

Arguments

data	finalized data from run_mab_trial().
bandits	Finalized bandits list from run_mab_trial().
algorithm	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
periods	Numeric scalar; total number of periods in Multi-Arm-Bandit trial.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

fix_negatives *Ensure Non-Zero Probabilities of Assignment*

Description

Redistributes Probabilities when control augmentation produces negatives

Usage

```
fix_negatives(assignment_probs, conditions, iter = 1)
```

Arguments

assignment_probs	Named Numeric Vector; Containing probabilities of treatment assignment
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
iter	iteration tracker, stops function if it reaches the limit of

Value

Named Numeric Vector; Containing probabilities of treatment assignment, all positive.

`get_adaptive_aipw` *Compute Adaptive AIPW Estimates of Treatment Success*

Description

Wrapper function around `get_iaipw()` and `adaptive_aipw()`. Computes Adaptive Augmented Inverse Probability Estimator for each treatment following formulation in Hadad et. al (2021).

Usage

```
get_adaptive_aipw(data, assignment_probs, periods, conditions, verbose)
```

Arguments

`data`	`final_data` object from `run_mab_trial()`. Contains results of Multi-Arm-Bandit Simulation
`assignment_probs`	`assignment_probs` object from `run_mab_trial()`. Contains probability of receiving each treatment at each treatment period in the simulated trial.
`periods`	Numeric; number of treatment waves.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.

Value

A named list containing:

`final_data`	`final_data` from `run_mab_trial()`, updated with individual AIPW scores]
`bandits`	Either the UCB1 statistics or Thompson Sampling posterior distributions.
`estimates`	object containing AIPW estimate and variance for each treatment

See Also

- `get_iaipw()`
- `adaptive_aipw()`
- Hadad et. al (2021)

get_bandit *Calculate Multi-Arm Bandit Decision Based on Algorithm*

Description

Calculates the best treatment for a given period using either a UCB1 or Thompson Sampling Algorithm.

Usage

```
get_bandit(
  past_results,
  algorithm,
  conditions,
  current_period = NULL,
  control_augment = 0
)
```

Arguments

past_results	data object containing summary of prior periods. Created by get_past_results().
algorithm	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
current_period	Numeric scalar; current period of the adaptive trial simulation.
control_augment	Number $\in$ [0,1]; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.

Value

The bandit object for the given period.

See Also

- run_mab_trial()
- get_past_results()

get_bandit.Thompson *Thompson Sampling Algorithm*

Description

Thompson Sampling Algorithm

Usage

```
## S3 method for class 'Thompson'
get_bandit(past_results, conditions, iterator)
```

Arguments

past_results	data object containing summary of prior periods. Created by get_past_results().
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
iterator	counter variable; keeps track of recursive calls to prevent infinite recursion.

Value

Named Numeric Vector of Posterior Probabilities

---

get_bandit.UCB1 *UCB1 Sampling Algorithm*

Description

UCB1 Sampling Algorithm

Usage

```
## S3 method for class 'UCB1'
get_bandit(past_results, conditions, current_period)
```

Arguments

past_results	data object containing summary of prior periods. Created by get_past_results().
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
current_period	Numeric scalar; current period of the adaptive trial simulation.

Value

Data.frame containing UCB and Success Rate for each condition

`get_iaipw` *Calculate Individual AIPW For Each Treatment Condition*

Description

Calculates the individual Augmented Inverse Probability Weighted Estimate (AIPW) of treatment success for each treatment condition provided.

Usage

```
get_iaipw(data, assignment_probs, periods, conditions, verbose)
```

Arguments

`data`	`final_data` object from `run_mab_trial()`. Contains results of Multi-Arm-Bandit Simulation
`assignment_probs`	`assignment_probs` object from `run_mab_trial()`. Contains probability of receiving each treatment at each treatment period in the simulated trial.
`periods`	Numeric; number of treatment waves.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.

Value

A data frame containing the data used in the MAB trial with new columns corresponding to the individual AIPW estimate for each treatment condition, and the probability of being assigned a given treatment condition.

See Also

- `run_mab_trial()`
- `get_adaptive_aipw()`
- `single_mab_simulation()`

`get_iaipw.data.frame` `get_iaipw()` *for data.frames*

Description

`get_iaipw()` for data.frames

Usage

```
## S3 method for class 'data.frame'
get_iaipw(data, assignment_probs, periods, conditions, verbose)
```

Arguments

data	final_data object from run_mab_trial(). Contains results of Multi-Arm-Bandit Simulation
assignment_probs	assignment_probs object from run_mab_trial(). Contains probability of receiving each treatment at each treatment period in the simulated trial.
periods	Numeric; number of treatment waves.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
verbose	Logical; Whether or not to print iteration number. FALSE by default.

---

get_iaipw.data.table *get_iaipw() for data.tables*

Description

get_iaipw() for data.tables

Usage

```
## S3 method for class 'data.table'
get_iaipw(data, assignment_probs, periods, conditions, verbose)
```

Arguments

data	final_data object from run_mab_trial(). Contains results of Multi-Arm-Bandit Simulation
assignment_probs	assignment_probs object from run_mab_trial(). Contains probability of receiving each treatment at each treatment period in the simulated trial.
periods	Numeric; number of treatment waves.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
verbose	Logical; Whether or not to print iteration number. FALSE by default.

---

get_past_results *Gather Past Results for Given Assignment Period*

Description

Summarizes results of prior periods to use for the current Multi-Arm-Bandit assignment.

Usage

```
get_past_results(
  current_data,
  prior_data,
  perfect_assignment,
  assignment_date_col = NULL,
  conditions
)
```

Arguments

current_data	Data with only observations from the current sampling period.
prior_data	Data with only the observations from the prior index.
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.
assignment_date_col	Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

Value

A data.frame, containing the number of successes, and number of people for each treatment condition.

See Also

*run_mab_trial() *single_mab_simulation() *get_bandit()

---

get_past_results.data.frame
get_past_results() for data.frames

Description

get_past_results() for data.frames

Usage

```
## S3 method for class 'data.frame'
get_past_results(
  current_data,
  prior_data,
  perfect_assignment,
  assignment_date_col = NULL,
  conditions
)
```

Arguments

current_data	Data with only observations from the current sampling period.
prior_data	Data with only the observations from the prior index.
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.
assignment_date_col	Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

---

get_past_results.data.table *get_past_results() for data.tables*

Description

get_past_results() for data.tables

Usage

```
## S3 method for class 'data.table'
get_past_results(
  current_data,
  perfect_assignment,
  assignment_date_col = NULL,
  conditions,
  prior_data
)
```

Arguments

current_data	Data with only observations from the current sampling period.
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.
assignment_date_col	Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE.
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
prior_data	Data with only the observations from the prior index.

`imputation_prep` *Prepping Data For Outcome Imputation*

Description

Pre-computes the conditional probabilities of success and dates of success for each distinct treatment block to impute them in `mab_simulation()` for those who get assigned new treatments.

Usage

```
imputation_prep(data, whole_experiment, perfect_assignment, data_cols)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`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.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

Value

A named list containing:

- `success`: Object the same type as `data`, which contains probability of success for each treatment block for each treatment period.
- `dates`: Average success date for each treatment block at each treatment period.

See Also

*`impute_success()` *`run_mab_trial()`

`imputation_prep.data.frame` *imputation Prep for data.frames*

Description

imputation Prep for data.frames

Usage

```
## S3 method for class 'data.frame'
imputation_prep(data, whole_experiment, perfect_assignment, data_cols)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`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.
`data_cols`	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 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'.
- `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
- `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

`imputation_prep.data.table` *imputation Prep for data.tables*

Description

imputation Prep for data.tables

Usage

```
## S3 method for class 'data.table'
imputation_prep(data, whole_experiment, perfect_assignment, data_cols)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`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.
`data_cols`	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 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'.
- `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
- `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

`impute_loop_prep` *Outcome imputation preparation*

Description

Prepares necessary data to be passed to`impute_success()` to impute outcomes properly. Subsets pre-computations and adds the `impute_block`, column to `current_data`

Usage

```
impute_loop_prep(
  current_data,
  block_cols,
  imputation_information,
  whole_experiment,
  blocking,
  perfect_assignment,
  current_period
)
```

Arguments

`current_data`	Data with only observations from the current sampling period.
`block_cols`	Character Vector of variables to block by.
`imputation_information`	Object created by `imputation_prep()` containing the conditional means and success dates for each treatment block to impute from.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`blocking`	Logical; Whether or not to use treatment blocking.
`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.
`current_period`	Numeric scalar; current treatment wave of the simulation.

Value

A named list containing:

- `current_data`: `data` object containing `impute_block` column to guide the outcome imputations
- `impute_success`: `data` object containing probabilities of success by `treatment_block` to be used to impute outcomes.
- `impute_dates`: Named Date Vector by treatment condition, containing the dates of success to impute if perfect_assignment is FALSE.

`impute_success` *Imputing New Outcomes under MAB Trial*

Description

Imputes Outcomes for the current treatment assignment period. Usesrandomizr::block_and_cluster_ra to impute success of the new treatments based on data from the original trial

Usage

```
impute_success(
  current_data,
  imputation_info,
  id_col,
  success_col,
  prior_data = NULL,
  perfect_assignment,
  dates = NULL,
  success_date_col,
  current_period = NULL
)
```

Arguments

`current_data`	Updated `data` object containing new treatments from `assign_treatments()` to impute outcomes for
`imputation_info`	`data` object containing probabilities of success from the original experiment, to impute outcomes from. Created by `imputation_prep()`
`id_col`	Column in data, contains unique id as a key.
`success_col`	Column in data; Binary successes from original experiment.
`prior_data`	`data` object from previous periods. Joined together at the end for the next iteration of the simulation.
`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.
`dates`	Named date vector; Contains average success date by treatment block to impute new success dates for observations whose change in treatment changes their outcome from failure to success.
`success_date_col`	Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
`current_period`	Numeric scalar; current treatment wave of the simulation.

See Also

- run_mab_trial()
- imputation_prep()
- randomizr::block_and_cluster_ra()
- randomizr::cluster_ra()

`impute_success.data.frame` *impute_success() for data.frames*

Description

impute_success() for data.frames

Usage

```
## S3 method for class 'data.frame'
impute_success(
  current_data,
  imputation_info,
  id_col,
  success_col,
  prior_data,
  perfect_assignment,
  dates = NULL,
  success_date_col,
  current_period
)
```

Arguments

current_data	Updated data object containing new treatments from assign_treatments() to impute outcomes for
imputation_info	data object containing probabilities of success from the original experiment, to impute outcomes from. Created by imputation_prep()
id_col	Column in data, contains unique id as a key.
success_col	Column in data; Binary successes from original experiment.
prior_data	data object from previous periods. Joined together at the end for the next iteration of the simulation.
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.
dates	Named date vector; Contains average success date by treatment block to impute new success dates for observations whose change in treatment changes their outcome from failure to success.

success_date_col
: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.

current_period
: Numeric scalar; current treatment wave of the simulation.

`impute_success.data.table` *`impute_success()` for data.tables*

Description

`impute_success()` for data.tables

Usage

```
## S3 method for class 'data.table'
impute_success(
  current_data,
  imputation_info,
  id_col,
  success_col,
  prior_data,
  perfect_assignment,
  dates = NULL,
  success_date_col,
  current_period
)
```

Arguments

current_data
: Updated data object containing new treatments from `assign_treatments()` to impute outcomes for

imputation_info
: data object containing probabilities of success from the original experiment, to impute outcomes from. Created by `imputation_prep()`

id_col
: Column in data, contains unique id as a key.

success_col
: Column in data; Binary successes from original experiment.

prior_data
: data object from previous periods. Joined together at the end for the next iteration of the simulation.

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.

dates
: Named date vector; Contains average success date by treatment block to impute new success dates for observations whose change in treatment changes their outcome from failure to success.

success_date_col
: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.

current_period
: Numeric scalar; current treatment wave of the simulation.

mab_simulation *Simulates Multi-Arm Bandit Trial From Prepared Inputs*

Description

Internal helper to `single_mab_simulation()` and `multiple_mab_simulation()`. Centralizes necessary functions to conduct a single Multi-Arm-Bandit Trial with adaptive inference. It assumes all inputs have been preprocessed by `pre_mab_simulation()`.

Usage

```
mab_simulation(
  data,
  time_unit,
  perfect_assignment,
  algorithm,
  period_length,
  prior_periods,
  whole_experiment,
  conditions,
  blocking,
  block_cols,
  data_cols,
  verbose,
  assignment_method,
  control_augment,
  imputation_information
)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`time_unit`	A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".
`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.
`algorithm`	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
`period_length`	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.
`prior_periods`	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.

`conditions` Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.

`blocking` Logical; Whether or not to use treatment blocking.

`block_cols` Character Vector of variables to block by.

`data_cols` 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 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'.
- `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
- `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

`verbose` Logical; Whether or not to print iteration number. FALSE by default.

`assignment_method` String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.

`control_augment` Number $\in [0,1]$; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.

`imputation_information` Object created by `imputation_prep()` containing the conditional means and success dates for each treatment block to impute from.

Value

`mab` class object, which is named list containing:

- `final_data`: The processed data with treatment assignments and imputed outcomes, labelled with "mab_" prefix.
- `bandits`: Either the UCB1 statistics or Thompson Sampling posterior distributions.
- `assignment_probs`: Probability of being assigned each treatment arm at a given period
- `estimates`: AIPW (Augmented Inverse Probability Weighting) treatment effect estimates and variances.
- `settings`: A list of the configuration settings used in the trial.

See Also

- `single_mab_simulation()`
- `multiple_mab_simulation()`
- `run_mab_trial()`
- `get_adaptive_aipw()`
- `pre_mab_simulation()`

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

Description

Repeated Multi-Arm Bandit Simulations with the same settings in different random states. Allows for parallel processing using `future::plan()` and `furrr::future_map()`.

Repeated Multi-Arm Bandit Simulations with the same settings in different random states. Allows for parallel processing using `future::plan()` and `furrr::future_map()`.

Usage

```
multiple_mab_simulation(
  data,
  assignment_method,
  algorithm,
  conditions,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  times,
  seeds,
  control_augment = 0,
  time_unit = NULL,
  period_length = NULL,
  block_cols = NULL,
  verbose = FALSE,
  keep_data = FALSE
)

multiple_mab_simulation(
  data,
  assignment_method,
  algorithm,
  conditions,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  times,
  seeds,
  control_augment = 0,
  time_unit = NULL,
  period_length = NULL,
  block_cols = NULL,
```

```
  verbose = FALSE,
  keep_data = FALSE
)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`assignment_method`	String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.
`algorithm`	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`prior_periods`	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
`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.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`blocking`	Logical; Whether or not to use treatment blocking.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.
`times`	Integer; number of simulations to conduct.
`seeds`	Integer vector of `length(times)` containing valid seeds to define random state for each trial.
`control_augment`	Number $\in [0,1]$; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.
`time_unit`	A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".

period_length	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified time_unit (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.
block_cols	Character Vector of variables to block by.
verbose	Logical; Toggles progress bar from furrr::future_map().
keep_data	Logical; Whether or not to keep the final data from each trial. Recommended FALSE for large datasets .

Value

multiple.mab class object, which is a named list containing:

- final_data_nest: Data.frame containing a nested data.frame with the final data from each trial
- bandits: Data.frame containing the Thompson/UCB1 statistics across all treatments, periods, and trials
- estimates: Data.frame containing the AIPW statistics across all treatments, and trials
- settings: A list of the configuration settings used in the trial.

multiple.mab class object, which is a named list containing:

- final_data_nest: Data.frame containing a nested data.frame with the final data from each trial
- bandits: Data.frame containing the Thompson/UCB1 statistics across all treatments, periods, and trials
- estimates: Data.frame containing the AIPW statistics across all treatments, and trials
- settings: A list of the configuration settings used in the trial.

See Also

- run_mab_trial()
- get_adaptive_aipw()
- check_args()
- single_mab_simulation()
- mab_simulation()
- pre_mab_simulation()
- furrr::future_map()
- future::plan()

- run_mab_trial()
- get_adaptive_aipw()
- check_args()
- single_mab_simulation()
- mab_simulation()
- pre_mab_simulation()
- furrr::future_map()
- future::plan()

Examples

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

# The seeds passed must be integers, so it is highly recommended to create them
# before using `sample.int()`
set.seed(1)
seeds <- sample.int(10000, 5)
conditions <- c("no_letter", "open_appt", "specific_appt")

# For this example, period_length is set a large interval and
# times is low to keep run time short.
start <- proc.time()
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,
  conditions = conditions,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success"
  ),
  verbose = FALSE, times = 5, seeds = seeds, keep_data = TRUE
)
seq_time <- proc.time() - start
print(x)

# Its Recommenced to set keep_data at FALSE unless necessary to avoid
# the output from taking up to much memory
# Keep TRUE
object.size(x)
x$final_data_nest <- NULL
# Size if Keep was FALSE
object.size(x)

# multiple_mab_simulation() is implemented using furrr::future_map()
# so you can also run simulations in parallel using futures.
# Simply run your preferred plan and number of cores before multiple_mab_simulation.
# Like:
## Not run:

  future::plan("plan", workers = n)
  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,
                          conditions = conditions,
                          data_cols = c(
                            condition_col = "condition",
                            id_col = "id",
                            success_col = "success"
                          ),
                          verbose = FALSE, times = 5, seeds = seeds, keep_data = TRUE
  )
  future::plan("sequential")

## End(Not run)
# If your on Windows plan needs to be multisession
# If your on Unix (MacOS/Linux) you can use multicore or multisession
# If your running the code on a high performance cluster, look into
# using the future.batchtools API for whichever scheduler is used

# Check the future and furrr documentation for more details on possible options
# Multiple_mab_simulation() is a useful tool for running multiple trials
# using the same configuration settings, in different random states
data(tanf)
# Subsetting to make the example faster
tanf <- tanf[1:50, ]

# The seeds passed must be integers, so it is highly recommended to create them
# before using `sample.int()`
set.seed(1)
seeds <- sample.int(10000, 5)
conditions <- c("no_letter", "open_appt", "specific_appt")

# For this example, period_length is set a large interval and
# times is low to keep run time short.
start <- proc.time()
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,
  conditions = conditions,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success"
  ),
  verbose = FALSE, times = 5, seeds = seeds, keep_data = TRUE
)
seq_time <- proc.time() - start
print(x)
```

```
# Its Recommenced to set keep_data at FALSE unless necessary to avoid
# the output from taking up to much memory
# Keep TRUE
object.size(x)
x$final_data_nest <- NULL
# Size if Keep was FALSE
object.size(x)

# multiple_mab_simulation() is implemented using furrr::future_map()
# so you can also run simulations in parallel using futures.
# Simply run your preferred plan and number of cores before multiple_mab_simulation.
# Like:
## Not run:

  future::plan("plan", workers = n)
  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,
                          conditions = conditions,
                          data_cols = c(
                            condition_col = "condition",
                            id_col = "id",
                            success_col = "success"
                          ),
                          verbose = FALSE, times = 5, seeds = seeds, keep_data = TRUE
  )
  future::plan("sequential")

## End(Not run)
# If your on Windows plan needs to be multisession
# If your on Unix (MacOS/Linux) you can use multicore or multisession
# If your running the code on a high performance cluster, look into
# using the future.batchtools API for whichever scheduler is used

# Check the future and furrr documentation for more details on possible options
```

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

Description

Uses `ggplot2::ggplot()` to summarize the results of a single Multi-Arm Bandit Trial

Usage

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

Arguments

x	mab class object created by single_mab_simulation()
type	String; Type of plot requested; valid types are: • arm: Shows Thompson Probability or UCB1 Statistic over the trial period. • assign: Shows Assignment Probability/Proportion over trial period. • estimate: Shows proportion of success estimates with user specified Normal Confidence Intervals based on their estimated variance.
estimator	Estimator to plot; Either "AIPW", "Sample" or "Both"; only used by "estimate" type
level	Confidence Interval Width (i.e 0.90, .95, 0.99)
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, etc.)

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 that has several minimal plots to examine
# the trial quickly
#
# These functions require ggplot2
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # 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",
    conditions = c(
      "no_letter",
      "open_appt",
      "specific_appt"
    ),
    data_cols = c(
      condition_col = "condition",
      id_col = "id",
      success_col = "success"
    )
  )

  # The plot generic has several options
  # specify type = arm, to plot the Thompson probabilities or UCB1 statistics
```

```
  # over the treatment periods of the trial
  y <- plot(x, type = "arm")
  y
  # These can be added to like any ggplot2 object

  y + ggplot2::labs(title = "Your New Title")

  # type = assign creates a similar plot, but its only useful when
  # control_augmentation is > 0.


  # Setting type = estiamte, allows for plotting of the
  # Augmented Inverse Probability Estimates.
  # By default it provides 95% Normal Confidence Intervals but this can be adjusted
  plot(x, type = "estimate", estimator = "AIPW")

  # Each type only uses 1 ggplot2 geom* so any arguments for the particular geom
  # can be added into the generic call
  # Changing the height for `geom_errorbarh`
  plot(x, type = "estimate", estimator = "AIPW", height = 0.4)
}
```

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

Description

Uses `ggplot2::ggplot()` to summarize the results of multiple Multi-Arm Bandit Trials

Usage

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

Arguments

x	`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.

- `estimate`: Shows proportion of success estimates user specified normal or empirical confidence intervals.

`estimator`	Estimator to plot; Either "AIPW", "Sample" or "Both"; used by `hist` and `estimate`.
`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`.
`level`	Confidence Interval Width (i.e 0.90, .95, 0.99)
`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.)

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
#
# # These functions require ggplot2
if (requireNamespace("ggplot2", quietly = TRUE)) {
  data(tanf)
  # Subsetting to make the example faster
  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,
    conditions = conditions,
    data_cols = c(
      condition_col = "condition",
      id_col = "id",
      success_col = "success"
    ),
    verbose = FALSE, times = 5, seeds = seeds, keep_data = FALSE
  )

  # The plot generic has several options
  # Specify type = summary, to get a bar graph showing each time
  # a treatment group was selected as the best.
  plot(x, type = "summary")
```

```
  # type = hist, creates a histogram of AIPW, Sample, or Both estimates for each
  # treatment over each trial
  plot(x, type = "hist", estimator = "AIPW")

  # type = estimate creates a similar error bar plot like in plot.mab()
  # but here the empirical variance of the estimate can be used instead
  plot(x, type = "estimate", estimator = "AIPW", cdf = "empirical")

  # These plots can be added to like any ggplot2 object
  plot(x, type = "summary") + ggplot2::labs(title = "Your New Title")

  # Each only uses 1 geom, so arguments for them can be added in the function call
  plot(x, type = "hist", estimator = "AIPW", binwidth = 0.05)
}
```

`plot_arms` *Plot Treatment Arms Over Time*

Description

Helper to `plot.mab()`. Plots Treatment Arms over Time.

Usage

```
plot_arms(x, object, ...)
```

Arguments

`x`	mab object passed from `plot.mab()`
`object`	String; Location to gather treatment arm data from, either "bandits" or "assignment_probs"
`...`	arguments to pass to `ggplot2:geom_*` function (e.g. `color`, `linewidth`, `alpha`, etc.)

Value

ggplot object

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

plot_estimates *Plot AIPW/Sample Estimates*

Description

Plot Summary of AIPW estimates and variances for Each Treatment Arm

Usage

```
plot_estimates(x, estimator, level = 0.95, ...)
```

Arguments

x	mab class object created by single_mab_simulation()
estimator	Estimator to plot; Either "AIPW", "Sample" or "Both"; only used by "estimate" type
level	Confidence Interval Width (i.e 0.90, .95, 0.99)
...	arguments to pass to ggplot2:geom_* function (e.g. color, linewidth, alpha, etc.)

Value

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

plot_hist *Plots Distribution of AIPW and Sample estimates over trials*

Description

Plots Distribution of AIPW and Sample estimates over trials for plot.multiple.mab()

Usage

```
plot_hist(x, estimator, ...)
```

Arguments

x	multiple.mab class object created by multiple_mab_simulation()
estimator	Estimator to plot; Either "AIPW", "Sample" or "Both"; used by hist and estimate.
...	arguments to pass to ggplot2:geom_* function (e.g. color, linewidth, alpha, bins etc.)

Value

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

plot_mult_estimates *Plots AIPW/Sample Estimates for each Arm*

Description

Plots AIPW/Sample Estimates for each arm using variance from the repeated trials.

Usage

```
plot_mult_estimates(x, estimator, cdf, level, ...)
```

Arguments

x	multiple.mab class object created by multiple_mab_simulation()
estimator	Estimator to plot; Either "AIPW", "Sample" or "Both"; used by hist and estimate.
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.
level	Confidence Interval Width (i.e 0.90, .95, 0.99)
...	arguments to pass to ggplot2:geom_* function (e.g. color, linewidth, alpha, bins etc.)

Value

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

plot_summary *Plot treatment Arms over multiple trials*

Description

Plots Summary Results for plot.multiple.mab()

Usage

```
plot_summary(x, ...)
```

Arguments

x	multiple.mab class object created by multiple_mab_simulation()
...	arguments to pass to ggplot2:geom_* function (e.g. color, linewidth, alpha, bins etc.)

Value

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

pre_mab_simulation *Pre-Simulation Setup for* mab_simulation()

Description

Common function for all the actions that need to take place before running the Multi-Arm-Bandit Simulation. Intakes data and column names, check for valid arguments, prepare data and pre-compute key values.

Usage

```
pre_mab_simulation(
  data,
  assignment_method,
  algorithm,
  conditions,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  control_augment,
  time_unit,
  period_length,
  block_cols,
  verbose
)
```

Arguments

data	A data frame, tibble or data.table that provides the input data for the trial.
assignment_method	String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.
algorithm	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
conditions	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
prior_periods	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
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.
whole_experiment	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
blocking	Logical; Whether or not to use treatment blocking.

`data_cols` 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 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'.
- `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
- `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

`control_augment` Number $\in$ [0,1]; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.

`time_unit` A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".

`period_length` Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

`block_cols` Character Vector of variables to block by.

`verbose` Logical; Whether or not to print iteration number. FALSE by default.

Value

Named list containing:

- `data_cols`: List of necessary columns in `data` as strings and symbols.
- `block_cols`: List of columns to block by in `data` as strings and symbols.
- `data`: Prepared `data` object the same class as inputted. Contains all columns required for `mab_simulation()`.
- `imputation_information`: List containing necessary information for outcome and date imputation for `mab_simulation()`.

See Also

*`single_mab_simulation()` *`multiple_mab_simulation()` *`check_args()` *`create_cutoff()` *`create_new_cols()` *`imputation_prep()`

print.mab *Print Generic For* mab

Description

Custom Print Display for objects of mab class returned by single_mab_simulation().

Usage

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

Arguments

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

Value

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

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

Description

Custom Print Display for 'multiple.mab" objects returned by multiple_mab_simulation().

Usage

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

Arguments

x	multiple.mab class object
...	further arguments passed to or from other methods

Value

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

print_mab *Print Helper for* mab *and* multiple.mab

Description

Common items for the print generics for mab and multiple.mab classes

Usage

```
print_mab(mab)
```

Arguments

mab	mab or multiple.mab object to derive settings from

Value

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

run_mab_trial *Runs Multi-Arm Bandit Trial*

Description

Performs a full Multi-Arm Bandit (MAB) trial using Thompson Sampling or UCB1. The function provides loop around each step of the process for each treatment wave, performing adaptive treatment assignment, and outcome imputation. Supports flexible customization in treatment blocking strategy, the size of each treatment wave, and information availability to simulate both a real experiment, and non-stationary bandit strategy.

Usage

```
run_mab_trial(
  data,
  time_unit,
  period_length = NULL,
  data_cols,
  block_cols,
  blocking,
  prior_periods,
  algorithm,
  whole_experiment,
  perfect_assignment,
  conditions,
  verbose,
  control_augment,
  imputation_information
)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`time_unit`	A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".
`period_length`	Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.
`data_cols`	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 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'. • `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'. • `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE. • `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.
`block_cols`	Character Vector of variables to block by.
`blocking`	Logical; Whether or not to use treatment blocking.
`prior_periods`	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
`algorithm`	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`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.
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`verbose`	Logical; Whether or not to print iteration number. FALSE by default.
`control_augment`	Number $\in [0,1]$; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.
`imputation_information`	Object created by `imputation_prep()` containing the conditional means and success dates for each treatment block to impute from.

Value

A named list containing:

- `final_data`: Processed data with new treatment assignments and imputed outcomes labelled with "mab_" prefix.
- `bandits`: Thompson Probability or UCB1 statistic for each treatment arm at each period of the simulation.
- `assignment_probs`: Assignment probabilities for each treatment arm at each period of the simulation.

See Also

- `single_mab_simulation()`
- `mab_simulation()`
- `create_prior()`
- `get_bandit()`
- `assign_treatments()`
- `check_impute()`
- `get_past_results()`
- `impute_success()`

---

single_mab_simulation *Running Multi-Arm Bandit Trial and Adaptive 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). This function wraps around `run_mab_trial()` and `get_adaptive_aipw()`, completing the full MAB pipeline: treatment assignment, success imputation, and estimation.

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). This function wraps around `run_mab_trial()` and `get_adaptive_aipw()`, completing the full MAB pipeline: treatment assignment, success imputation, and estimation.

Usage

```
single_mab_simulation(
  data,
  assignment_method,
  algorithm,
  conditions,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  control_augment = 0,
```

```
  time_unit = NULL,
  period_length = NULL,
  block_cols = NULL,
  verbose = FALSE
)

single_mab_simulation(
  data,
  assignment_method,
  algorithm,
  conditions,
  prior_periods,
  perfect_assignment,
  whole_experiment,
  blocking,
  data_cols,
  control_augment = 0,
  time_unit = NULL,
  period_length = NULL,
  block_cols = NULL,
  verbose = FALSE
)
```

Arguments

`data`	A data frame, tibble or data.table that provides the input data for the trial.
`assignment_method`	String; "Date", "Batch" or "Individual" to define the assignment into treatment waves.
`algorithm`	A string specifying the MAB algorithm to use. Options are "Thompson" or "UCB1".
`conditions`	Named Character vector containing treatment conditions. Control condition, must be named "Control" when 'control_augment' > 0.
`prior_periods`	Numeric; number of previous periods to use in the treatment assignment model or specify string "All" to use all previous periods.
`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.
`whole_experiment`	Logical; if TRUE, uses all past experimental data for imputing outcomes. If FALSE, uses only data available up to the current period.
`blocking`	Logical; Whether or not to use treatment blocking.
`data_cols`	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 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'.
- `month_col`: Column in data, contains month of treatment; only necessary when time_unit = 'Month'.
- `success_date_col`: Column in data, contains original dates each success occurred; only necessary when 'perfect_assignment' = FALSE.
- `assignment_date_col`: Column in data, contains original dates treatments are assigned to observations; only necessary when 'perfect_assignment' = FALSE. Used to simulate imperfect information on part of researchers conducting an adaptive trial.

`control_augment`
: Number $\in$ [0,1]; Proportion of each wave guaranteed to get "Control" treatment. Default is 0.

`time_unit`
: A string specifying the unit of time for assigning periods when 'assignment_method' is 'date'. Acceptable values are "Day", "Week", or "Month".

`period_length`
: Numeric; length of each treatment period. If assignment method is "Date", this refers to the length of periods by your specified `time_unit` (i.e., if "Day", 10 would be 10 days). If assignment methods is "Batch", this refers to the number of people in each batch.

`block_cols`
: Character Vector of variables to block by.

`verbose`
: Logical; Whether or not to print iteration number. FALSE by default.

Value

`mab` class object, which is named list containing:

- `final_data`: The processed data with treatment assignments and imputed outcomes, labelled with "mab_" prefix.
- `bandits`: Either the UCB1 statistics or Thompson Sampling posterior distributions.
- `assignment_probs`: Probability of being assigned each treatment arm at a given period
- `estimates`: AIPW (Augmented Inverse Probability Weighting) treatment effect estimates and variances.
- `settings`: A list of the configuration settings used in the trial.

`mab` class object, which is named list containing:

- `final_data`: The processed data with treatment assignments and imputed outcomes, labelled with "mab_" prefix.
- `bandits`: Either the UCB1 statistics or Thompson Sampling posterior distributions.
- `assignment_probs`: Probability of being assigned each treatment arm at a given period
- `estimates`: AIPW (Augmented Inverse Probability Weighting) treatment effect estimates and variances.
- `settings`: A list of the configuration settings used in the trial.

See Also

- `run_mab_trial()`
- `get_adaptive_aipw()`
- `check_args()`

- mab_simulation()
- pre_mab_simulation()
- run_mab_trial()
- get_adaptive_aipw()
- check_args()
- mab_simulation()
- pre_mab_simulation()

Examples

```
# Loading Example Data and defining conditions
data(tanf)
conditions <- c("no_letter", "open_appt", "specific_appt")

## 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,
  conditions = conditions,
  perfect_assignment = FALSE,
  data_cols = c(
    condition_col = "condition",
    id_col = "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
names(conditions) <- c("Control", "T1", "T2")
# 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,
  conditions = conditions,
  control_augment = 0.25,
```

```
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success",
    date_col = "appt_date",
    month_col = "recert_month"
  )
)

## If you misspecify or miss an argument, an appropriate error will be given
## I specified Month assignment but did not provide a month_column in my data

try(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 = FALSE,
  conditions = conditions,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success",
    date_col = "appt_date"
  )
))

# I specified a negative period_length

try(single_mab_simulation(
  data = tanf,
  assignment_method = "Date",
  time_unit = "Month",
  algorithm = "UCB1",
  period_length = -500,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = FALSE,
  conditions = conditions,
  control_augment = 0,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success",
    date_col = "appt_date"
  )
))

# I forgot to add column containing the successes of the original experiment

try(single_mab_simulation(
  data = tanf,
```

```
  assignment_method = "Batch",
  algorithm = "Thompson",
  period_length = 500,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  conditions = conditions,
  control_augment = 0,
  data_cols = c(
    condition_col = "condition",
    id_col = "service_center"
  )
))
# Loading Example Data and defining conditions
data(tanf)
conditions <- c("no_letter", "open_appt", "specific_appt")

## 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,
  conditions = conditions,
  perfect_assignment = FALSE,
  data_cols = c(
    condition_col = "condition",
    id_col = "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
names(conditions) <- c("Control", "T1", "T2")
# 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,
  conditions = conditions,
```

```
  control_augment = 0.25,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success",
    date_col = "appt_date",
    month_col = "recert_month"
  )
)

## If you misspecify or miss an argument, an appropriate error will be given
## I specified Month assignment but did not provide a month_column in my data

try(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 = FALSE,
  conditions = conditions,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success",
    date_col = "appt_date"
  )
))

# I specified a negative period_length

try(single_mab_simulation(
  data = tanf,
  assignment_method = "Date",
  time_unit = "Month",
  algorithm = "UCB1",
  period_length = -500,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = FALSE,
  conditions = conditions,
  control_augment = 0,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success",
    date_col = "appt_date"
  )
))

# I forgot to add column containing the successes of the original experiment

try(single_mab_simulation(
```

```
  data = tanf,
  assignment_method = "Batch",
  algorithm = "Thompson",
  period_length = 500,
  prior_periods = "All",
  blocking = FALSE,
  whole_experiment = TRUE,
  perfect_assignment = TRUE,
  conditions = conditions,
  control_augment = 0,
  data_cols = c(
    condition_col = "condition",
    id_col = "service_center"
  )
))
```

---

summary.mab *Summary Generic for "mab" class*

Description

Summarizes the Results of a Single Multi-Arm Bandit Trial.

Usage

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

Arguments

object	"mab" class object created by single_mab_simulation().
level	Confidence Interval Width (i.e 0.90, .95, 0.99)
...	additional arguments.

Value

data.frame containg each treatment, the final Thompson/UCB1 Statistic, the AIPW estimate and Normal CI based on user supplied level.

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",
  conditions = c(
    "no_letter",
    "open_appt",
    "specific_appt"
  ),
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success"
  )
)

# Calling `summary` Returns a summary table for the trial
# Defaults to 95% Normal Confidence Intervals
# for the Augmented Inverse Probability Estimates
summary(x)

# We can also change the confidence level to anything between 0 and 1
summary(x, level = 0.7)

# Invalid levels throw an error
try(summary(x, level = 5))
```

`summary.multiple.mab` *Summary Generic for "multiple.mab" class*

Description

Summarizes results of of multiple Multi-Arm Bandit Trials

Usage

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

Arguments

`object`	`multiple.mab` object created by multiple_mab_simulation
`level`	Confidence Interval Width (i.e 0.90, .95, 0.99)
`...`	additional arguments.

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)
# Subsetting to make the example faster
tanf <- tanf[1:100, ]
# Simulating a few trials

seeds <- sample.int(10000, 5)
```

```
conditions <- c("no_letter", "open_appt", "specific_appt")
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,
  conditions = conditions,
  data_cols = c(
    condition_col = "condition",
    id_col = "id",
    success_col = "success"
  ),
  verbose = FALSE, times = 5, seeds = seeds, keep_data = FALSE
)

# Calling `summary` Returns a summary table for the trial
# Upper and Lower Bounds default to 95% Confidence Intervals
summary(x) |>
  print(width = Inf) # calling width = Inf to so whole table prints

# We can also change the confidence level to anything between 0 and 1
# This only changes the upper and lower bounds that are presented.
summary(x, level = 0.7) |>
  dplyr::select(lower_normal:upper_empirical)

# Invalid levels throw an error
try(summary(x, level = 5))
```

`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 3520 rows and 22 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

---

`verbose_log` *Verbose Printer*

Description

Shorthand Function for checking `verbose` and then printing. Takes verbose from higher scope

Usage

```
verbose_log(log, message)
```

Arguments

`log`	Logical; Whether or not to print the message
`message`	The message to be printed to screen, as a string.

Index