Package 'banditsCI'

May 7, 2026

Title Bandit-Based Experiments and Policy Evaluation

Version 1.0.0

Description Frequentist inference on adaptively generated data. The methods implemented are based on Zhan et al. (2021) <doi:10.48550/arXiv.2106.02029> and Hadad et al. (2021) <doi:10.48550/arXiv.191 lustration, several functions for simulating non-contextual and contextual adaptive experiments using Thompson sampling are also supplied.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/UChicago-pol-methods/banditsCI, https://uchicago-pol-methods.github.io/banditsCI/

BugReports https://github.com/UChicago-pol-methods/banditsCI/issues

Suggests knitr (>= 1.43), rmarkdown (>= 2.23), testthat (>= 3.0.0)

Config/testthat/edition 3

Imports glmnet (>= 4.1-6), MASS (>= 7.3-56), mvtnorm (>= 1.2-2), Rdpack (>= 2.6)

RdMacros Rdpack

Collate 'experiment_utils.R' 'adaptive_utils.R' 'errors.R'

VignetteBuilder knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Molly Offer-Westort [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2787-9919>), Yinghui Zhou [aut] (ORCID: <https://orcid.org/0009-0002-3515-9751>), Ruohan Zhan [aut] (ORCID: <https://orcid.org/0000-0002-3426-2784>)

Maintainer Molly Offer-Westort <mollyow@gmail.com>

Repository CRAN

Date/Publication 2024-11-29 09:20:02 UTC

Contents

`.check_A` *Check Number of Observations for Inference*

Description

This function checks if the number of observations is greater than 1, which is required for conducting inference.

Usage

```
.check_A(A)
```

Arguments

`A`	An integer representing the number of observations.

Value

Returns NULL if the number of observations is valid; otherwise, throws an error.

.check_first_batch *Check First Batch Validity*

Description

This function checks if the first batch size is greater than or equal to the number of treatment arms.

Usage

```
.check_first_batch(batch_sizes, ys)
```

Arguments

batch_sizes	A numeric vector specifying batch sizes.
ys	A matrix of counterfactual conditions.

Value

Returns NULL if the first batch size is valid; otherwise, throws an error.

.check_shape *Check Shape Compatibility of Probability Objects*

Description

This function checks the dimensional compatibility of ‘gammahats‘ and ‘contextual_probs‘. It validates the dimensions and types based on whether the probability objects are contextual or non-contextual.

Usage

```
.check_shape(gammahats, contextual_probs)
```

Arguments

gammahats	A matrix representing estimates.
contextual_probs	An object representing probabilities, either a matrix (non-contextual) or an array (contextual).

Value

Returns NULL if the shapes and types are valid; otherwise, throws an error.

`aw_estimate` *Estimate policy value via non-contextual adaptive weighting.*

Description

Estimates the value of a policy based on AIPW scores and a policy matrix using non-contextual adaptive weighting. If `evalwts` is not provided, uses equal weights for all observations.

Usage

```
aw_estimate(scores, policy, evalwts = NULL)
```

Arguments

`scores`	Numeric matrix. AIPW scores, shape `[A, K]`, where `A` is the number of observations and `K` is the number of arms. Must not contain NA values.
`policy`	Numeric matrix. Policy matrix $\pi(X_t, w)$, shape `[A, K]`. Must have the same shape as `scores` and must not contain NA values.
`evalwts`	Optional numeric vector. Non-contextual adaptive weights h_t, length `A`, or `NULL`. Default is `NULL`.

Value

Numeric scalar. Estimated policy value.

Examples

```
scores <- matrix(c(0.5, 0.8, 0.6,
                   0.3, 0.9, 0.2,
                   0.5, 0.7, 0.4,
                   0.8, 0.2, 0.6), ncol = 3, byrow = TRUE)
policy <- matrix(c(0.2, 0.3, 0.5,
                   0.6, 0.1, 0.3,
                   0.4, 0.5, 0.1,
                   0.2, 0.7, 0.1), ncol = 3, byrow = TRUE)
aw_estimate(scores = scores, policy = policy, evalwts = c(0.5, 1, 0.5, 1.5))
```

aw_scores *Compute AIPW/doubly robust scores.*

Description

Computes AIPW/doubly robust scores based on observed rewards, pulled arms, and inverse probability scores. If `mu_hat` is provided, compute AIPW scores, otherwise compute IPW scores.

Usage

```
aw_scores(yobs, ws, balwts, K, mu_hat = NULL)
```

Arguments

`yobs`	Numeric vector. Observed rewards. Must not contain NA values.
`ws`	Integer vector. Pulled arms. Must not contain NA values. Length must match `yobs`.
`balwts`	Numeric matrix. Inverse probability score $1[W_t = w]/e_t(w)$ of pulling arms, shape `[A, K]`, where `A` is the number of observations and `K` is the number of arms. Must not contain NA values.
`K`	Integer. Number of arms. Must be a positive integer.
`mu_hat`	Optional numeric matrix. Plug-in estimator of arm outcomes, shape `[A, K]`, or `NULL`. Must not contain NA values if provided.

Value

Numeric matrix. AIPW scores, shape `[A, K]`.

Examples

```
aw_scores(yobs = c(0.5, 1, 0, 1.5),
          ws = c(1, 2, 2, 3),
          balwts = matrix(c(0.5, 2, 1, 0.5,
                            1, 1.5, 0.5, 1.5,
                            2, 1.5, 0.5, 1),
                            ncol = 3),
          K = 3,
          mu_hat = matrix(c(0.5, 0.8, 0.6, 0.3,
                            0.9, 0.2, 0.5, 0.7,
                            0.4, 0.8, 0.2, 0.6),
                            ncol = 3))
```

aw_var *Variance of policy value estimator via non-contextual adaptive weighting.*

Description

Computes the variance of a policy value estimate based on AIPW scores, a policy matrix, and non-contextual adaptive weights.

Usage

```
aw_var(scores, estimate, policy, evalwts = NULL)
```

Arguments

scores	Numeric matrix. AIPW scores, shape [A, K]. Must not contain NA values.
estimate	Numeric scalar. Policy value estimate.
policy	Numeric matrix. Policy matrix $\pi(X_t, w)$, shape [A, K]. Must have the same shape as scores and must not contain NA values.
evalwts	Optional numeric vector. Non-contextual adaptive weights h_t, length A, or NULL.

Value

Numeric scalar. Variance of policy value estimate.

Examples

```
scores <- matrix(c(0.5, 0.8, 0.6,
                   0.3, 0.9, 0.2,
                   0.5, 0.7, 0.4,
                   0.8, 0.2, 0.6), ncol = 3, byrow = TRUE)
policy <- matrix(c(0.2, 0.3, 0.5,
                   0.6, 0.1, 0.3,
                   0.4, 0.5, 0.1,
                   0.2, 0.7, 0.1), ncol = 3, byrow = TRUE)
estimate <- aw_estimate(scores = scores, policy = policy, evalwts = c(0.5, 1, 0.5, 1.5))
aw_var(scores = scores, estimate = estimate, policy = policy, evalwts = c(0.5, 1, 0.5, 1.5))
```

calculate_balwts *Calculate balancing weight scores.*

Description

Calculates the inverse probability score of pulling arms, given the actions taken and the true probabilities of each arm being chosen.

Usage

```
calculate_balwts(ws, probs)
```

Arguments

ws	Integer vector. Indicates which arm was chosen for observations at each time t. Length A. Must not contain NA values.
probs	Numeric matrix or array. True probabilities of each arm being chosen at each time step. Shape [A, K] or [A, A, K]. Must not contain NA values.

Value

A matrix or array containing the inverse probability score of pulling arms.

Examples

```
set.seed(123)
A <- 5
K <- 3
ws <- sample(1:K, A, replace = TRUE)
probs <- matrix(runif(A * K), nrow = A, ncol = K)
balwts <- calculate_balwts(ws, probs)
```

calculate_continuous_X_statistics *Estimate/variance of policy evaluation via contextual weighting.*

Description

Computes the estimate and variance of a policy evaluation based on adaptive weights, AIPW scores, and a policy matrix.

Usage

```
calculate_continuous_X_statistics(h, gammahat, policy)
```

Arguments

h	Numeric matrix. Adaptive weights $h_t(X_s)$, shape [A, A]. Must be a square matrix and must not contain NA values.
gammahat	Numeric matrix. AIPW scores, shape [A, K]. Must not contain NA values.
policy	Numeric matrix. Policy matrix $\pi(X_t, w)$, shape [A, K]. Must have the same shape as gammahat and must not contain NA values.

Value

Named numeric vector with elements estimate and var, representing the estimated policy value and the variance of the estimate, respectively.

Examples

```
h <- matrix(c(0.4, 0.3, 0.2, 0.1,
              0.2, 0.3, 0.3, 0.2,
              0.5, 0.3, 0.2, 0.1,
              0.1, 0.2, 0.1, 0.6), ncol = 4, byrow = TRUE)
scores <- matrix(c(0.5, 0.8, 0.6, 0.3,
                   0.9, 0.2, 0.5, 0.7,
                   0.4, 0.8, 0.2, 0.6), ncol = 3, byrow = TRUE)
policy <- matrix(c(0.2, 0.3, 0.5,
                   0.6, 0.1, 0.3,
                   0.4, 0.5, 0.1,
                   0.2, 0.7, 0.1), ncol = 3, byrow = TRUE)
gammahat <- scores - policy
calculate_continuous_X_statistics(h = h, gammahat = gammahat, policy = policy)
```

---

draw_thompson *Thompson Sampling draws.*

Description

Draws arms from a LinTS or non-contextual TS agent for multi-armed bandit problems.

Usage

```
draw_thompson(model, start, end, xs = NULL)
```

Arguments

model	List. Contains the parameters of the model, generated by LinTSModel().
start	Integer. Starting index of observations for which arms are to be drawn. Must be a positive integer.
end	Integer. Ending index of the observations for which arms are to be drawn. Must be an integer greater than or equal to start.
xs	Optional matrix. Covariates of shape [A, p], where p is the number of features, if the LinTSModel is contextual. Default is NULL. Must not contain NA values.

Value

A list containing the drawn arms (w) and their corresponding probabilities (ps).

Examples

```
set.seed(123)
model <- LinTSModel(K = 5, p = 3, floor_start = 1/5, floor_decay = 0.9, num_mc = 100,
                    is_contextual = TRUE)
draws <- draw_thompson(model = model, start = 1, end = 10,
                       xs = matrix(rnorm(30), ncol = 3))
```

---

estimate *Estimate/variance of policy evaluation via non-contextual weighting.*

Description

Computes the estimate and variance of a policy evaluation based on non-contextual weights, AIPW scores, and a policy matrix.

Usage

```
estimate(w, gammahat, policy)
```

Arguments

w	Numeric vector. Non-contextual weights, length A. Must not contain NA values.
gammahat	Numeric matrix. AIPW scores, shape [A, K]. Must not contain NA values.
policy	Numeric matrix. Policy matrix $\pi(X_t, w)$, shape [A, K]. Must have the same shape as gammahat and must not contain NA values.

Value

Named numeric vector with elements estimate and var, representing the estimated policy value and the variance of the estimate, respectively.

Examples

```
w <- c(0.5, 1, 0.5, 1.5)
scores <- matrix(c(0.5, 0.8, 0.6,
                   0.3, 0.9, 0.2,
                   0.5, 0.7, 0.4,
                   0.8, 0.2, 0.6), ncol = 3, byrow = TRUE)
policy <- matrix(c(0.2, 0.3, 0.5,
                   0.6, 0.1, 0.3,
                   0.4, 0.5, 0.1,
                   0.2, 0.7, 0.1), ncol = 3, byrow = TRUE)
gammahat <- scores - policy
```

```
estimate(w = w, gammahat = gammahat,
policy = policy)
```

`generate_bandit_data` *Generate classification data.*

Description

Generates covariates and potential outcomes for a classification dataset.

Usage

```
generate_bandit_data(xs = NULL, y = NULL, noise_std = 1, signal_strength = 1)
```

Arguments

`xs`	Optional matrix. Covariates of shape `[A, p]`, where `A` is the number of observations and `p` is the number of features. Default is `NULL`. Must not contain NA values.
`y`	Optional vector. Labels of length `A`. Default is `NULL`. Must not contain NA values.
`noise_std`	Numeric. Standard deviation of the noise added to the potential outcomes. Default is `1.0`. Must be a non-negative number.
`signal_strength`	Numeric. Strength of the signal in the potential outcomes. Default is `1.0`.

Value

A list containing the generated data (`xs`, `ys`, `muxs`, `A`, `p`, `K`) and the true class probabilities (`mus`).

Examples

```
data <- generate_bandit_data(xs = as.matrix(iris[,1:4]),
                             y = as.numeric(iris[,5]),
                             noise_std = 0.1,
                             signal_strength = 1.0)
```

ifelse_clip *Clip lamb values between a minimum x and maximum y.*

Description

Clips a numeric vector between two values.

Usage

```
ifelse_clip(lamb, x, y)
```

Arguments

lamb	Numeric vector. Values to be clipped.
x	Numeric. Lower bound of the clip range.
y	Numeric. Upper bound of the clip range.

Value

Numeric vector. Clipped values.

Examples

```
lamb <- c(1, 2, 3, 4, 5)
ifelse_clip(lamb, 2, 4)
```

impose_floor *Impose probability floor.*

Description

Imposes a floor on the given array a, ensuring that its elements are greater than or equal to amin.

Usage

```
impose_floor(a, amin)
```

Arguments

a	Numeric vector. Must not contain NA values.
amin	Numeric. Minimum allowed value. Must be between 0 and 1.

Value

A numeric vector with the same length as a, with the floor imposed on its elements.

Examples

```
a <- c(0.25, 0.25, 0.25, 0.25)
imposed_a <- impose_floor(a = a, amin = 0.1)
```

`LinTSModel` *Linear Thompson Sampling model.*

Description

Creates a linear Thompson Sampling model for multi-armed bandit problems.

Usage

```
LinTSModel(
  K,
  p = NULL,
  floor_start,
  floor_decay,
  num_mc = 100,
  is_contextual = TRUE
)
```

Arguments

`K`	Integer. Number of arms. Must be a positive integer.
`p`	Integer. Dimension of the contextual vector, if `is_contextual` is set to `TRUE`. Otherwise, `p` is ignored. Must be a positive integer.
`floor_start`	Numeric. Specifies the initial value for the assignment probability floor. It ensures that at the start of the process, no assignment probability falls below this threshold. Must be a positive number.
`floor_decay`	Numeric. Decay rate of the floor. The floor decays with the number of observations in the experiment such that at each point in time, the applied floor is: `floor_start/(s^{floor_decay})`, where `s` is the starting index for a batched experiment, or the observation index for an online experiment. Must be a number between 0 and 1 (inclusive).
`num_mc`	Integer. Number of Monte Carlo simulations used to approximate the expected reward. Must be a positive integer. Default is 100.
`is_contextual`	Logical. Indicates whether the problem is contextual or not. Default is `TRUE`.

Value

A list containing the parameters of the LinTSModel.

Examples

```
model <- LinTSModel(K = 5, p = 3, floor_start = 1/5, floor_decay = 0.9, num_mc = 100,
                    is_contextual = TRUE)
```

`output_estimates` *Policy evaluation with adaptively generated data.*

Description

Calculates average response and differences in average response under counterfactual treatment policies. Estimates are produced using provided inverse probability weighted (IPW) or augmented inverse probability weighted (AIPW) scores paired with various adaptive weighting schemes, as proposed in Hadad et al. (2021) and Zhan et al. (2021).

We briefly outline the target quantities: For observations indexed $t \in \{1, \ldots, A\}$, treatments $w \in \{1, \ldots, K\}$, we denote as $Y_t(w)$ the potential outcome for the unit at time t under treatment w. A policy π is a treatment assignment procedure that is the subject of evaluation, described in terms of treatment assignment probabilities for each subject to receive each counterfactual treatment. We target estimation of average response under a specified policy:

$$Q(\pi) := \sum_{w=1}^{K} \mathrm{E}\left[\pi(w) Y_t(w)\right]$$

The user may specify a list of list of policies to be evaluated, under `policy1`.

Alternatively, they may estimate policy contrasts if `policy0` is provided:

$$\Delta(\pi^1, \pi^2) := Q(\pi^1) - Q(\pi^2)$$

Usage

```
output_estimates(
  policy0 = NULL,
  policy1,
  contrasts = "combined",
  gammahat,
  probs_array,
  uniform = TRUE,
  non_contextual_minvar = TRUE,
  contextual_minvar = TRUE,
  non_contextual_stablevar = TRUE,
  contextual_stablevar = TRUE,
  non_contextual_twopoint = TRUE,
  floor_decay = 0
)
```

Arguments

policy0	Optional matrix. Single policy probability matrix for contrast evaluation, dimensions [A, K]. Each row represents treatment assignment probabilities for an individual subject, and so rows must sum to 1. When policy0 = NULL, the function estimates the value $Q(\pi)$ of each policy matrix listed in policy1. When policy0 is non-null, the function estimates differences in average response under each of the component policies in policy1 and the *single* policy in policy0. Must not contain NA values if provided.
policy1	List of matrices. List of counterfactual policy matrices for evaluation, dimensions [A, K]. Each row represents treatment assignment probabilities for an individual subject, and so rows must sum to 1. Must not contain NA values.
contrasts	Character. The method to estimate policy contrasts, either combined or separate, discussed in Hadad et al. (2021) Section 3. combined indicates the difference in (A)IPW scores is directly used as the unbiased scoring rule for $\Delta(\pi^1, \pi^2)$; separate indicates that scores are used separately $\hat{\Delta}(\pi^1, \pi^2) = \hat{Q}(w_1) - \hat{Q}(w_2)$.
gammahat	(A)IPW scores matrix with dimensions [A, K] in non-contextual settings, or [A, A, K] contextual settings. Dimensions represent time, (contexts,) treatment arms. Dimensions of gammahat and probs_array must be the same. Must not contain NA values.
probs_array	Numeric array. Probability matrix or array with dimensions [A, K] in non-contextual settings, or [A, A, K] contextual settings. Dimensions represent time, (contexts,) treatment arms. Dimensions of gammahat and probs_array must be the same. Must not contain NA values.
uniform	Logical. Estimate uniform weights.
non_contextual_minvar	Logical. Estimate non-contextual MinVar weights described in Zhan et al. (2021) Section 4.
contextual_minvar	Logical. Estimate contextual MinVar weights described in Zhan et al. (2021) Section 4.
non_contextual_stablevar	Logical. Estimate non-contextual StableVar weights described in Zhan et al. (2021) Section 4.
contextual_stablevar	Logical. Estimate contextual StableVar weights described in Zhan et al. (2021) Section 4.
non_contextual_twopoint	Logical. Estimate two-point allocation weights described in Hadad et al. (2021) Section 2.
floor_decay	Numeric. Floor decay parameter used in the calculation. Default is 0.

Value

A list of treatment effect estimates under different weighting schemes.

References

Hadad V, Hirshberg DA, Zhan R, Wager S, Athey S (2021). "Confidence intervals for policy evaluation in adaptive experiments." *Proceedings of the national academy of sciences*, **118**(15), e2014602118.

Zhan R, Hadad V, Hirshberg DA, Athey S (2021). "Off-policy evaluation via adaptive weighting with data from contextual bandits." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2125–2135.

Examples

```
set.seed(123)
# In a non-contextual setting, generate example values for policy1, gammahat, and probs_array
gammahat <- matrix(c(0.5, 0.8, 0.6,
                     0.3, 0.9, 0.2,
                     0.5, 0.7, 0.4,
                     0.8, 0.2, 0.6), ncol = 3, byrow = TRUE)
policy0 <- matrix(c(1, 0, 0,
                    1, 0, 0,
                    1, 0, 0,
                    1, 0, 0), ncol = 3, byrow = TRUE)
policy1 <- list(matrix(c(0, 1, 0,
                         0, 1, 0,
                         0, 1, 0,
                         0, 1, 0), ncol = 3, byrow = TRUE))

probs_array <- array(0, dim = c(4, 4, 3))
for (i in 1:4) {
  temp_vector <- runif(3)
  normalized_vector <- temp_vector / sum(temp_vector)
  probs_array[i, 1, ] <- normalized_vector
}
for (k in 1:3) {
  for (i in 1:4) {
    temp_vector <- runif(3)
    normalized_vector <- temp_vector / sum(temp_vector)
    probs_array[i, 2:4, k] <- normalized_vector
  }
}
estimates <- output_estimates(policy1 = policy1,
                              policy0 = policy0,
                              gammahat = gammahat,
                              probs_array = probs_array)
# plot
plot_results <- function(result) {
  estimates <- result[, "estimate"]
  std.errors <- result[, "std.error"]
  labels <- rownames(result)

  # Define the limits for the x-axis based on estimates and std.errors
  xlims <- c(min(estimates - 2*std.errors), max(estimates + 2*std.errors))
```

```
  # Create the basic error bar plot using base R
  invisible(
    plot(estimates, 1:length(estimates), xlim = xlims, xaxt = "n",
         xlab = "Coefficient Estimate", ylab = "",
         yaxt = "n", pch = 16, las = 1, main = "Coefficients and CIs")
  )

  # Add y-axis labels
  invisible(
    axis(2, at = 1:length(estimates), labels = labels, las = 1, tick = FALSE,
         line = 0.5)
  )

  # Add the x-axis values
  x_ticks <- x_ticks <- seq(from = round(xlims[1], .5),
                            to = round(xlims[2], .5), by = 0.5)
  invisible(
    axis(1,
         at = x_ticks,
         labels = x_ticks)
  )

  # Add error bars
  invisible(
    segments(estimates - std.errors,
             1:length(estimates),
             estimates + std.errors,
             1:length(estimates))
  )
}

sample_result <- estimates[[1]]
op <- par(no.readonly = TRUE)
par(mar=c(5, 12, 4, 2))
plot_results(sample_result)
par(op)
```

`plot_cumulative_assignment` *Plot cumulative assignment for bandit experiment.*

Description

Generates a plot of the cumulative assignment.

Usage

```
plot_cumulative_assignment(results, batch_sizes)
```

Arguments

results	List. Results of the experiment, including the actions taken (ws) and the true probabilities of each arm being chosen (probs).
batch_sizes	Integer vector. Batch sizes used in the experiment. Must be positive integers.

Value

A plot of the cumulative assignment.

Examples

```
set.seed(123)
A <- 1000
K <- 4
xs <- matrix(runif(A * K), nrow = A, ncol = K)
ys <- matrix(rbinom(A * K, 1, 0.5), nrow = A, ncol = K)
batch_sizes <- c(250, 250, 250, 250)
results <- run_experiment(ys = ys,
                          floor_start = 1/K,
                          floor_decay = 0.9,
                          batch_sizes = batch_sizes,
                          xs = xs)
plot_cumulative_assignment(results, batch_sizes)
```

---

ridge_init *Ridge Regression Initialization for Arm Expected Rewards*

Description

Initializes matrices needed for ridge regression to estimate the expected rewards of different arms.

Usage

```
ridge_init(p, K)
```

Arguments

p	Integer. Number of covariates. Must be a positive integer.
K	Integer. Number of arms. Must be a positive integer.

Value

A list containing initialized matrices R_A, R_Ainv, b, and theta for each arm.

Examples

```
p <- 3
K <- 5
init <- ridge_init(p, K)
```

`ridge_muhat_lfo_pai` *Leave-future-out ridge-based estimates for arm expected rewards.*

Description

Computes leave-future-out ridge-basedn estimates of arm expected rewards based on provided data.

Usage

```
ridge_muhat_lfo_pai(xs, ws, yobs, K, batch_sizes, alpha = 1)
```

Arguments

`xs`	Matrix. Covariates of shape `[A, p]`, where `A` is the number of observations and `p` is the number of features. Must not contain NA values.
`ws`	Integer vector. Indicates which arm was chosen for observations at each time `t`. Length `A`. Must not contain NA values.
`yobs`	Numeric vector. Observed outcomes, length `A`. Must not contain NA values.
`K`	Integer. Number of arms. Must be a positive integer.
`batch_sizes`	Integer vector. Sizes of batches in which data is processed. Must be positive integers.
`alpha`	Numeric. Ridge regression regularization parameter. Default is 1.

Value

A 3D array containing the expected reward estimates for each arm and each time `t`, of shape `[A, A, K]`.

Examples

```
set.seed(123)
p <- 3
K <- 5
A <- 100
xs <- matrix(runif(A * p), nrow = A, ncol = p)
ws <- sample(1:K, A, replace = TRUE)
yobs <- runif(A)
batch_sizes <- c(25, 25, 25, 25)
muhat <- ridge_muhat_lfo_pai(xs, ws, yobs, K, batch_sizes)
print(muhat)
```

`ridge_update` *Updates ridge regression matrices.*

Description

Given previous matrices and a new observation, updates the matrices for ridge regression.

Usage

```
ridge_update(R_A, b, xs, t, yobs, alpha)
```

Arguments

`R_A`	Matrix. Current matrix `R_A` for an arm. Must not contain NA values.
`b`	Numeric vector. Current vector `b` for an arm.
`xs`	Matrix. Covariates of shape `[A, p]`, where `A` is the number of observations and `p` is the number of features. Must not contain NA values.
`t`	Integer. Current time or instance.
`yobs`	Numeric vector. Observed outcomes, length `A`. Must not contain NA values.
`alpha`	Numeric. Ridge regression regularization parameter. Default is 1.

Value

A list containing updated matrices `R_A`, `R_Ainv`, `b`, and `theta`.

Examples

```
set.seed(123)
p <- 3
K <- 5
init <- ridge_init(p, K)
R_A <- init$R_A[[1]]
b <- init$b[1, ]
xs <- matrix(runif(10 * p), nrow = 10, ncol = p)
yobs <- runif(10)
t <- 1
alpha <- 1
updated <- ridge_update(R_A, b, xs, t, yobs[t], alpha)
```

`run_experiment` *Run an experiment using Thompson Sampling.*

Description

Runs a LinTS or non-contextual TS bandit experiment, given potential outcomes and covariates.

Usage

```
run_experiment(
  ys,
  floor_start,
  floor_decay,
  batch_sizes,
  xs = NULL,
  balanced = NULL
)
```

Arguments

`ys`	Matrix. Potential outcomes of shape `[A, K]`, where `A` is the number of observations and `K` is the number of arms. Must not contain NA values.
`floor_start`	Numeric. Specifies the initial value for the assignment probability floor. It ensures that at the start of the process, no assignment probability falls below this threshold. Must be a positive number.
`floor_decay`	Numeric. Decay rate of the floor. The floor decays with the number of observations in the experiment such that at each point in time, the applied floor is: `floor_start/(s^{floor_decay})`, where `s` is the starting index for a batched experiment, or the observation index for an online experiment. Must be a number between 0 and 1 (inclusive).
`batch_sizes`	Integer vector. Size of each batch. Must be positive integers.
`xs`	Optional matrix. Covariates of shape `[A, p]`, where `p` is the number of features, if the LinTSModel is contextual. Default is `NULL`. Must not contain NA values.
`balanced`	Optional logical. Indicates whether to balance the batches. Default is `NULL`.

Value

A list containing the pulled arms (`ws`), observed rewards (`yobs`), assignment probabilities (`probs`), and the fitted bandit model (`fitted_bandit_model`).

Examples

```
set.seed(123)
A <- 1000
K <- 4
xs <- matrix(runif(A * K), nrow = A, ncol = K)
```

```
ys <- matrix(rbinom(A * K, 1, 0.5), nrow = A, ncol = K)
batch_sizes <- c(250, 250, 250, 250)
results <- run_experiment(ys = ys,
                          floor_start = 1/K,
                          floor_decay = 0.9,
                          batch_sizes = batch_sizes,
                          xs = xs)
```

`simple_tree_data` *Generate simple tree data.*

Description

Generates covariates and potential outcomes of a synthetic dataset for a simple tree model.

Usage

```
simple_tree_data(
  A,
  K = 5,
  p = 10,
  noise_std = 1,
  split = 1.676,
  signal_strength = 1,
  noise_form = "normal"
)
```

Arguments

`A`	Integer. Number of observations in the dataset. Must be a positive integer.
`K`	Integer. Number of arms. Must be a positive integer.
`p`	Integer. Number of covariates. Must be a positive integer.
`noise_std`	Numeric. Standard deviation of the noise added to the potential outcomes. Must be a non-negative number.
`split`	Numeric. Split point for creating treatment groups based on the covariates.
`signal_strength`	Numeric. Strength of the signal in the potential outcomes.
`noise_form`	Character. Distribution of the noise added to the potential outcomes. Can be either "normal" or "uniform".

Value

A list containing the generated data (`xs`, `ys`, `muxs`) and the true potential outcome means (`mus`).

Examples

```
set.seed(123)
A <- 1000
K <- 4     # Number of treatment arms
p <- 10    # Number of covariates
synthetic_data <- simple_tree_data(A = A,
                                   K = K,
                                   p = p,
                                   noise_std = 1.0,
                                   split = 1.676,
                                   signal_strength = 1.0,
                                   noise_form = 'normal')
```

`stick_breaking` *Stick breaking function.*

Description

Implements the stick breaking algorithm for calculating weights in the stable-var scheme.

Usage

```
stick_breaking(Z)
```

Arguments

`Z`	Numeric array. Input array, shape `[A, K]`. Must not contain NA values.

Value

Numeric array. Stick breaking weights, shape `[A, K]`. Must not contain NA values.

Examples

```
set.seed(123)
Z <- array(runif(10), dim = c(2, 5))
stick_breaking(Z)
```

twopoint_stable_var_ratio *Calculate allocation ratio for a two-point stable-variance bandit.*

Description

Calculates the allocation ratio for a two-point stable-variance bandit, given the empirical mean and the discount parameter alpha.

Usage

```
twopoint_stable_var_ratio(A, e, alpha)
```

Arguments

A	Integer. Size of the experiment. Must be a positive integer.
e	Numeric. Empirical mean. Must be a numeric value.
alpha	Numeric. Discount parameter.

Value

Numeric vector. Allocation ratio lambda.

Examples

```
# Calculate the allocation ratio for a two-point stable-variance bandit with e=0.1 and alpha=0.5
twopoint_stable_var_ratio(1000, 0.1, 0.5)
```

update_thompson *Update linear Thompson Sampling model.*

Description

Updates the parameters of a linear Thompson Sampling model for multi-armed bandit problems based on new observations.

Usage

```
update_thompson(ws, yobs, model, xs = NULL, ps = NULL, balanced = NULL)
```

Arguments

ws	Integer vector. Indicates which arm was chosen for observations at each time t. Length A, where A is the number of observations. Must not contain NA values.
yobs	Numeric vector. Observed outcomes, length A. Must not contain NA values.
model	List. Contains the parameters of the LinTSModel.
xs	Optional matrix. Covariates of shape [A, p], where p is the number of features, if the LinTSModel is contextual. Default is NULL. Must not contain NA values.
ps	Optional matrix. Probabilities of selecting each arm for each observation, if the LinTSModel is balanced. Default is NULL.
balanced	Logical. Indicates whether to use balanced Thompson Sampling. Default is NULL.

Value

A list containing the updated parameters of the LinTSModel.

Examples

```
set.seed(123)
model <- LinTSModel(K = 5, p = 3, floor_start = 1, floor_decay = 0.9, num_mc = 100,
                    is_contextual = TRUE)
A <- 1000
ws <- numeric(A)
yobs <- numeric(A)
model <- update_thompson(ws = ws, yobs = yobs, model = model)
```

Index