Package 'latentState'

September 17, 2026

Title Simulate Outcomes of a Latent State Reinforcement Learning Model

Version 1.0.0

Description Simulates outcomes of an updated version of the latent state reinforcement learning model originally described in Cochran and Cisler (2019) <doi:10.1371/journal.pcbi.1007331>. The package is designed to create results under all reasonable experiment setups, including different reinforcement schedules, number of cues, number of phases, and number of options per trial. Participants can be simulated using either fixed parameters or parameters drawn from a distribution.

License GPL (>= 3)

URL https://osf.io/2whcu

Encoding UTF-8

RoxygenNote 8.0.0

NeedsCompilation no

Author Martin Benada [aut, cre]

Maintainer Martin Benada <martinibenada@gmail.com>

Repository CRAN

Date/Publication 2026-09-17 09:30:09 UTC

Contents

latentState-package *latentState: Simulate Outcomes of a Latent State Reinforcement Learning Model*

Description

Simulates outcomes of an updated version of the latent state reinforcement learning model originally described in Cochran and Cisler (2019) doi:10.1371/journal.pcbi.1007331. The package is designed to create results under all reasonable experiment setups, including different reinforcement schedules, number of cues, number of phases, and number of options per trial. Participants can be simulated using either fixed parameters or parameters drawn from a distribution.

Author(s)

Maintainer: Martin Benada <martinibenada@gmail.com>

Authors:

- Martin Benada <martinibenada@gmail.com>

See Also

Useful links:

- https://osf.io/2whcu

make_world *Build a fixed, non-randomized trial world for use with* run()

Description

Constructs the c_vec (cue array) and win (reward array) needed for the my_world argument of run(), letting you specify an exact trial-by-trial cue and reward sequence instead of having one randomly generated from phase_def.

Usage

```
make_world(trial_patterns, reward, L = 10, features = NULL)
```

Arguments

trial_patterns	Character vector, one element per trial, giving the cues shown on each arm. Use letters for cues, "_" to separate arms, and "0" for an empty (unselectable) arm – the same naming convention used for trial-type names in phase_def (e.g. "A", "AX_B", "A_0"). Every element must have the same number of arms.

`reward` Numeric vector, or list of numeric vectors. Gives the reward value for each trial, applied identically to every arm on that trial regardless of which arm is later chosen by the agent (unlike `phase_def`, where reward probabilities can differ by cue/arm). A plain numeric vector is treated as a single reward dimension (`D = 1`); to specify multiple reward dimensions, supply a list of numeric vectors, one per dimension. Every vector must have length equal to `length(trial_patterns)`.

`L` Integer. The total number of latent states the resulting `c_vec` is built for. Must match the total number of latent states used in the `run()` call this world is passed into (i.e. `ncop`), since `run()` validates `my_world`'s dimensions against that value. Defaults to `10`.

`features` Character vector, or `NULL`. Lets you restrict which features (cues) are tracked, instead of the default of every individual letter plus every full multi-letter arm segment appearing in `trial_patterns`. Each entry must be composed of letters that co-occur together in at least one arm segment, and every arm segment in `trial_patterns` must be covered by at least one feature. This mirrors the `features` argument of `run()`, but validated against `trial_patterns` directly. Defaults to `NULL` (features generated automatically).

Value

A list with three elements, matching the format `run()` expects for `my_world`:

`c_vec` Array (`D x C x L x A x ntrials`) of 0/1 cue activations, indicating which cues are active on each arm and trial, identical across every latent state.

`win` Array (`D x A x ntrials`) of reward values, with the same value broadcast across every arm on a given trial.

`type_seq` A one-element list wrapping `trial_patterns`, matching the phase-structured format `run()` produces internally.

Examples

```
my_world <- make_world(
  trial_patterns = c("A", "A", "B", "A", "B"),
  reward         = c(1, 0, 1, 1, 0),
  L = 10
)
phase_def <- list(phase1 = list(trials = list(A = 3, B = 2)))
result <- run(phase_def, n = 1, my_world = my_world)
```

---

plot_compare *Compare latent-state learning results across groups*

Description

Overlays results from multiple `run()` calls (e.g. different groups, conditions, or parameter settings) on the same plots, allowing direct visual comparison. Cues are distinguished by color and groups by line type (or by color alone, for the presentation-based plots).

Usage

```
plot_compare(
  ...,
  plot = FALSE,
  plot_avg = FALSE,
  fixed_axis = TRUE,
  plot_sd = FALSE,
  plot_label = FALSE,
  plot_shown = FALSE,
  plot_cue = FALSE,
  plot_appear = FALSE
)
```

Arguments

`...`	One or more outputs of `run()`, i.e. lists of participant results, passed as named arguments (e.g. `ctrl = result1, treatment = result2`). The names become group labels used in plot legends; if left unnamed, groups are labeled `"group1"`, `"group2"`, etc. All groups must share the same task structure (`par$D`, `par$C`, `par$base_cues`, and trial/phase layout), since the first group's structure is used as the reference for all plots.
`plot`	Logical. If `TRUE`, plots V across trials for participant 1 of each group, overlaid on the same axes – cues are colored, groups are distinguished by line type. Combine with `plot_label` to add trial-type labels (colored by reward) on the x-axis, using the reference group's trial sequence. Defaults to `FALSE`.
`plot_avg`	Logical. If `TRUE`, plots the mean V across participants within each group, across trials, overlaid on the same axes – cues are colored, groups are distinguished by line type. Combine with `plot_sd` to add a shaded standard-deviation ribbon per group. Defaults to `FALSE`.
`fixed_axis`	Logical. If `TRUE`, y-axes are fixed to `[-0.5, 0.5]`, keeping plots comparable across cues and groups. If `FALSE`, each plot's y-axis is scaled to its own data range. Defaults to `TRUE`.
`plot_sd`	Logical. If `TRUE`, adds a shaded ribbon showing +/- 1 SD around each group's mean line, for any group with more than one participant. Applies to `plot_avg`, `plot_cue`, `plot_shown`, and `plot_appear`. Defaults to `FALSE`.
`plot_label`	Logical. If `TRUE`, adds trial-type labels (colored by reward) along the x-axis of the `plot` output, taken from the reference group's trial sequence. Requires every participant in every group to share an identical trial sequence. Defaults to `FALSE`.
`plot_shown`	Logical. If `TRUE`, plots V by presentation number for each distinct trial-type cue combination (e.g. `"A"`, `"AB"`), comparing the mean across participants for each group on the same axes, colored by group. Combine with `plot_sd` to add a shaded standard-deviation ribbon. Defaults to `FALSE`.
`plot_cue`	Logical. If `TRUE`, plots each individual base cue's own value by presentation number, pooling every trial type that cue appears in, comparing the mean across participants for each group on the same axes, colored by group. Combine with `plot_sd` to add a shaded standard-deviation ribbon. Defaults to `FALSE`.

plot_appear — Logical. If TRUE, plots the total value of the chosen arm by presentation number for each base cue, restricted to presentations where that cue was part of the chosen arm, comparing the mean across participants for each group on the same axes, colored by group. Combine with plot_sd to add a shaded standard-deviation ribbon. Defaults to FALSE.

Value

No return value. Called for its side effect of producing one or more base R plots comparing latent-state values across the supplied groups. Invisibly returns NULL.

Examples

```
phase_def <- list(phase1 = list(trials = list(A = 20, B = 20)))
ctrl      <- run(phase_def, n = 5, alpha0 = "0.1")
treatment <- run(phase_def, n = 5, alpha0 = "0.3")
plot_compare(ctrl = ctrl, treatment = treatment, plot_avg = TRUE)
```

---

run *Simulate latent-state reinforcement learning across participants*

Description

Simulates outcomes of an updated latent-state reinforcement learning model (based on Cochran & Cisler, 2019) across one or more participants, under a user-specified experimental design. For full mathematical details of the underlying latent-state learning model, including changes from the original Cochran & Cisler model, see the supplementary materials on OSF: https://osf.io/2whcu

Usage

```
run(
  phase_def,
  n = 1,
  seed = 42,
  alpha0 = "0.2",
  alpha1 = "0.05",
  alpha2 = "0.05",
  gamma = "0.01",
  eta = "1.2",
  delta = "0.6",
  sigma0 = "0.5",
  tau = "10",
  lambda = "1",
  pseudo = NULL,
  block = NULL,
  probabilistic = FALSE,
```

```
  features = NULL,
  V_noise_sd = 0,
  ncop = 10,
  group = "ctrl",
  prev = 0,
  ITI = NULL,
  ezITI = NULL,
  chi = "1",
  contextShift = NULL,
  my_world = NULL,
  initialex = NULL,
  plot = FALSE,
  plot_all = FALSE,
  plot_label = FALSE,
  plot_arm = FALSE,
  plot_mu = FALSE,
  plot_mu_all = FALSE,
  plot_shown = FALSE,
  plot_cue = FALSE,
  plot_sd = FALSE,
  plot_raw = FALSE,
  plot_raw_all = FALSE,
  plot_q = FALSE,
  plot_q_all = FALSE,
  plot_sigma = FALSE,
  plot_sigma_all = FALSE,
  plot_appears = FALSE,
  fixed_axis = TRUE
)
```

Arguments

phase_def	A named list defining the task structure. Each element represents one phase of the experiment and must contain two entries, `trials` and (optionally) `rewards`. `trials` is a named list where each entry is a cue label (e.g. `"A"`, `"AB"`) and its value is the number of trials of that type in the phase. Each letter represents a unique cue, and combining multiple letters (e.g. `"AB"`) means those cues are shown simultaneously on the same arm. `rewards` is a named list with one entry per reward dimension (e.g. `"D1"`, `"D2"`). Each entry is a named numeric vector mapping cue labels to reward probabilities between 0 and 1. Only cues that appear in `trials` may be listed in `rewards`. By default, probabilities are applied deterministically: exactly that proportion of trials of that type are rewarded, rather than each trial being rewarded independently at random. This requires each probability to divide evenly into the trial count. E.g. 0.75 requires a trial count divisible by 4. To reward trials independently at random instead, without this divisibility requirement, set `probabilistic = TRUE`. A simple example using a single reward dimension and one option per trial is

shown below. In phase 1, cue A appears alone on 6 trials and together with cue B on another 6 trials; A is rewarded on every trial, while B is omitted from rewards and is therefore automatically assigned a reward probability of 0. In phase 2, cue A appears alone on 8 trials and together with B on 16 trials; since rewards is omitted entirely for this phase, all trials are automatically unrewarded.

```
phase_def <- list(
  phase1 = list(
    trials  = list(A = 6, AB = 6),
    rewards = list(c(A = 1))
  ),
  phase2 = list(
    trials = list(A = 8, AB = 16)
  )
)
```

In experiments with multiple arms (choices), indicate the cues shown on each arm within a trial type name, separating arms with _. For example, a three-arm trial where one arm shows cues A and B, one shows only A, and one shows only C would be named "AB_A_C". All trial types within a phase must have the same number of arms. If you want a trial type to have fewer effective arms, mark an arm as empty using "0"; that arm is assigned a value of -Inf and can never be chosen. This can be used to force participants to select a specific option on certain trials – for example, "A_0_0" means only the arm with cue A can be chosen on that trial.

See **Examples** below for a phase_def that uses multiple arms per trial, empty ("0") arms, and multiple reward dimensions.

n — Integer. The number of participants to simulate. Each participant is run independently through the full task defined by phase_def, with parameters either fixed or freshly sampled (if specified as a distribution) for each participant. Defaults to 1.

seed — Integer. A random seed passed to set.seed() at the start of the simulation, ensuring that results (including parameter sampling, trial randomization, and probabilistic choices) are reproducible across runs. Defaults to 42.

alpha0 — Character string. The learning rate controlling how quickly associative strengths (V) update in response to prediction error. Must evaluate to a numeric value in [0, 1]. Passed as a string so it can specify either a fixed value (e.g. alpha0 = "0.2") or a distribution to sample from independently for each participant (e.g. alpha0 = "rbeta(1, 20, 80)"). Defaults to "0.2".

alpha1 — Character string. The learning rate controlling how quickly the variance estimate (sigmasq) updates based on squared prediction error. Must evaluate to a numeric value in [0, 1]. Passed as a string so it can specify either a fixed value (e.g. alpha1 = "0.05") or a distribution to sample from independently for each participant (e.g. alpha1 = "rbeta(1, 20, 80)"). Defaults to "0.05".

alpha2 — Character string. The learning rate controlling how quickly the effort matrix (B), which governs cue associability, updates toward the current belief-weighted cue structure. Must evaluate to a numeric value in [0, 1]. Passed as a string so

it can specify either a fixed value (e.g. `alpha2 = "0.05"`) or a distribution to sample from independently for each participant (e.g. `alpha2 = "rbeta(1, 20, 80)"`). Defaults to `"0.05"`.

`gamma` Character string. Controls the rate at which latent-state beliefs drift toward a uniform distribution over active states between trials. This same drift also determines the prior belief assigned to a newly created latent state: when a new state is added, it is initialized with a prior belief of `gamma`. Must evaluate to a numeric value in `[0, 1]`, where higher values cause beliefs to drift toward uniform more quickly (and give more initial credence to newly created states). Passed as a string so it can specify either a fixed value (e.g. `gamma = "0.01"`) or a distribution to sample from independently for each participant (e.g. `gamma = "rbeta(1, 20, 80)"`). Defaults to `"0.01"`.

`eta` Character string. The change-point threshold: a new latent state is created once the accumulated change-point statistic reaches or exceeds this value. Must evaluate to a non-negative numeric value, where higher values make new-state creation less likely (requiring stronger, sustained evidence of a change before splitting off a new state). Passed as a string so it can specify either a fixed value (e.g. `eta = "1.2"`) or a distribution to sample from independently for each participant (e.g. `eta = "rgamma(1, shape = 25, rate = 50)"`). Defaults to `"1.2"`.

`delta` Character string. A penalty that makes it harder for a new latent state to be created. Each trial, this penalty is subtracted from the evidence accumulating for a new state; if the evidence that trial isn't strong enough to outweigh the penalty, the accumulated evidence for a new state will decrease rather than increase. Higher values raise the bar for what counts as strong enough evidence, so with a large `delta`. Must evaluate to a non-negative numeric value. Passed as a string so it can specify either a fixed value (e.g. `delta = "0.6"`) or a distribution to sample from independently for each participant (e.g. `delta = "rgamma(1, shape = 25, rate = 50)"`). Defaults to `"0.6"`.

`sigma0` Character string. The agent's initial assumption about how much variability in outcomes is "expected," used to judge whether a prediction error is surprising enough to be considered evidence for a new state. A higher `sigma0` means the agent starts out expecting more noisy, unpredictable outcomes, so it takes larger prediction errors to be seen as informative. The agent's estimate of `sigma` will evolve throughout the simulation, so `sigma0` is mainly influential at the beginning of the simulation. Must evaluate to a positive numeric value. Passed as a string so it can specify either a fixed value (e.g. `sigma0 = "0.5"`) or a distribution to sample from independently for each participant (e.g. `sigma0 = "rgamma(1, shape = 25, rate = 50)"`). Defaults to `"0.5"`.

`tau` Character string. The inverse temperature parameter controlling how deterministic choices are. Higher values make the agent more likely to consistently pick the option with the highest expected value, while lower values make choices more random, giving lower-valued options a better chance of being picked anyway. Must evaluate to a positive numeric value. Passed as a string so it can specify either a fixed value (e.g. `tau = "10"`) or a distribution to sample from independently for each participant (e.g. `tau = "rgamma(1, shape = 25, rate = 50)"`). Defaults to `"10"`.

`lambda` Character string. Controls how much more strongly beliefs update on trials that

are especially informative about which latent state is active. Informativeness is measured by how unevenly the evidence favors one state over the others on that trial. When one state clearly stands out, `lambda` amplifies how much beliefs shift toward it; when the evidence is ambiguous across states, `lambda` has little effect. Must evaluate to a non-negative numeric value, where higher values produce larger belief updates on informative trials. Passed as a string so it can specify either a fixed value (e.g. `lambda = "1"`) or a distribution to sample from independently for each participant (e.g. `lambda = "rgamma(1, shape = 25, rate = 50)"`). Defaults to `"1"`.

`pseudo` Integer or `NULL`. Controls pseudorandomization of trial order within each phase: no more than `pseudo` consecutive trials may be of the same type. For example, `pseudo = 2` ensures the same trial type never appears more than twice in a row. If `NULL`, trial order is fully randomized with no such constraint. Ignored for any phase with only one trial type, since no reordering is possible. Defaults to `NULL`.

`block` Integer, numeric vector, or `NULL`. The number of trials in each block, combining all trial types in their correct proportions. For example, if a phase has two trial types in a 1:3 ratio and `block = 8`, each block will contain exactly 2 of the first type and 6 of the second, shuffled independently within that block. This produces a more even spread of trial types than randomizing order across the whole phase at once. A single number applies the same block length to every phase. A vector specifies a different length per phase. It must have one entry per phase. The number of trials in a phase must be evenly divisible by the block length, and each trial type's count must be evenly divisible by the resulting number of blocks. When `block` is used, `pseudo` is ignored. Defaults to `NULL`.

`probabilistic` Logical. If `TRUE`, trial types are drawn independently according to their specified probabilities, rather than fixing the exact count of each type in advance. For example, with two trial types in a 1:3 ratio and 100 trials, the default (fixed-count) approach always produces exactly 25 and 75 trials. With `probabilistic = TRUE`, each trial is instead assigned independently with a 25% and 75% chance, so realized counts vary across phases. Defaults to `FALSE`.

`features` Character vector, or `NULL`. Lets you restrict which features the agent tracks, instead of using every possible one. If `NULL` (default), features are generated automatically. Every single letter (e.g. `"A"`) becomes its own feature, and every multi-letter arm segment (e.g. `"AB"`) additionally becomes its own feature. For example, a trial with arm segment `"AB"` activates three features: `"A"`, `"B"`, and `"AB"`, meaning the agent tracks the AB combination as a distinct cue on top of tracking A and B individually.

To track only a subset of these – for example, dropping `"AB"` as its own feature so it's represented only through `"A"` and `"B"`, or dropping `"A"` as its own feature so it's only ever represented as part of a larger combination like `"AB"` – supply a character vector specifying exactly the features you want. Each entry becomes exactly one feature, and no others are created. Defaults to `NULL`.

`V_noise_sd` Numeric. Standard deviation of Gaussian noise added to the agent's associative strengths (`V`) on each trial, for plotting and output purposes only. The underlying `V` used for the agent's actual learning and choice behavior on future trials remains noise-free. The noisy version is computed separately each trial and only affects the overall `V` values recorded in the output (e.g. `V`, `V_cue`, `V_shown`,

V_appear), simulating measurement or observation noise without altering the agent's internal state. Noisy values are clamped to stay within [-0.5, 0.5], the valid range for centered reward values. Defaults to 0 (no noise).

ncop — Integer >= 1. Maximum number of latent states the agent can create. Default is 10.

group — Character. Label for this batch's experimental group (e.g. "ctrl", "treatment"), stored in the group column of the long-format output (long). Useful for combining multiple run calls into one dataset for ANOVA. Defaults to "ctrl".

prev — Integer. Offset added to subject IDs in the long-format output (long), so participants are numbered starting after prev instead of at 1. Prevents subject ID collisions when combining multiple run calls (e.g. across groups) into one dataset for ANOVA. For example, if a control group of 20 is run first, a subsequent call would use prev = 20 so IDs start at 21. Defaults to 0.

ITI — List of numeric vectors, or NULL. Timestamps (not gaps) marking when each trial occurs. Must be a list of length n (one vector per participant, even if n = 1), where each vector has length ntrials and is non-decreasing. This requires specifying a timestamp for every trial, even those with the default gap of 1; if only a few trials have a non-default gap, and that gap is the same for every participant, see ezITI instead, which only requires listing those exceptions.Gaps between consecutive timestamps partially decay latent-state beliefs toward uniform (with a decay of gamma for each one unit gap), simulating forgetting, and allow more rumination iterations (chi) on the preceding trial. Cannot be used together with ezITI.

For example, with ntrials = 3 and n = 2, ITI = list(c(1, 2, 3), c(1, 2, 10)) gives participant 1 the standard single gamma decay between each trial, while participant 2 has a large gap before trial 3, producing the equivalent of 8 successive gamma decays (since the gap between timestamps 2 and 10 is 8). Defaults to NULL (trials evenly spaced, one unit apart).

ezITI — List, or NULL. A simpler alternative to ITI for specifying uneven trial timing, shared across all participants (use ITI instead if you need per-participant timing). Each element is a length-2 numeric vector c(trial, gap), setting the gap immediately after that trial to gap timestamp units instead of the default gap of 1. All unspecified gaps remain 1. trial must be a whole number between 1 and ntrials - 1, and gap must be greater than 1. These gaps are used to build the full ITI timestamp vector internally, so they affect belief decay and rumination (chi) the same way ITI does. Cannot be used together with ITI.

For example, with ntrials = 3, ezITI = list(c(2, 8)) sets the gap after trial 2 to 8, producing the same effective timestamps as ITI = list(c(1, 2, 10)). This is equivalent to 8 successive gamma decays before trial 3. Defaults to NULL.

chi — Character. The number of "rumination" iterations the agent performs on a single trial's outcome before moving to the next trial. Higher values let the agent update its associative strengths repeatedly from the same outcome, simulating extra "thinking time" between trials. The actual number of iterations used is capped by the ITI gap before the next trial, so chi only has an effect on trials where that gap exceeds 1; if ITI is NULL, chi has no effect at all. Passed as a string, so it can specify either a fixed value (e.g. chi = "1") or a distribution to

sample from independently for each participant (e.g. chi = "rpois(1, 2) + 1"). Must be a whole number of at least 1. Defaults to "1" (no extra rumination).

contextShift — Numeric vector, or NULL. Trial numbers marking the first trial of a new context, at which latent-state beliefs are fully reset to uniform right before that trial's update, simulating a change in context (e.g. a new room). Unlike ITI, which gradually decays beliefs based on timestamp gaps, this resets them instantly. Only belief proportions (lsb) are affected – latent states, V, B, and sigmasq are untouched.

For example, contextShift = c(50, 100) resets beliefs right before trials 50 and 100. Defaults to NULL (no forced resets).

my_world — List, or NULL. Input a fixed, non-randomized order of cues and rewards. This will override the random order that is usually created from phase_def. This allows complete control over the trial sequence of the simulated experiment. The input should be built with the make_world() function, which returns the required c_vec(cue order array), win (reward order vector), and type_seq (flattened cue order vector) elements in the correct format. phase_def is still needed to determine trial counts, phase boundaries, and reward validation, but no new trial order or reward assignment is generated. Defaults to NULL (world is generated randomly).

initialex — Numeric array, or NULL. Sets the agent's initial associative strengths (V), overriding the default of 0. Must be an array of dimension D x C x L (dimensions x cues x latent states). initialex[d, c, l] is cue c's starting value on dimension d in state l. Only state 1 is active initially, so setting all L slices the same applies that value to every future state as it's created, while setting only slice 1 (rest at 0) leaves new states starting fresh at 0. Values must be in [-0.5, 0.5] (the centered reward scale). Defaults to NULL (every cue starts at 0 in every state).

plot — Logical. If TRUE, produces a plot of overall associative strength (V) across trials, with one line per cue. Phase boundaries are marked with dashed vertical lines. By default, only participant 1 is plotted; to plot every participant instead, use plot_all. Combine with plot_label to show trial-type labels (colored by reward) on the x-axis, and with plot_arm to also show which arm was chosen on each trial (when there's more than one option). Defaults to FALSE.

plot_all — Logical. Same as plot, but produces the V plot for every participant instead of only participant 1. Defaults to FALSE.

plot_label — Logical. If TRUE, adds trial-type labels along the x-axis of the relevant plots (plot/plot_all, plot_raw/plot_raw_all, plot_mu/plot_mu_all, plot_q/plot_q_all, plot_sigma/plot_sigma_all), colored green if that trial was rewarded and red if it wasn't. Defaults to FALSE.

plot_arm — Logical. If TRUE, adds the chosen arm number beneath the trial-type labels on the x-axis, for any of the following plots: plot/plot_all, plot_raw/plot_raw_all, plot_mu/plot_mu_all, plot_q/plot_q_all, plot_sigma/plot_sigma_all. Has no effect unless plot_label = TRUE. Defaults to FALSE.

plot_mu — Logical. If TRUE, plots the expected value (mu) of the chosen arm across trials, one plot per reward dimension, for participant 1. Combine with plot_label to show trial-type labels (colored by reward) on the x-axis, and plot_arm to also show the chosen arm beneath each label. Note: this compound's true value can

fall below -0.5 under some conditions (see Supplementary Materials S1.8.1). When it does, the plotted value is clamped to -0.5, which is the value of a non-rewarded trial and thus the minimum theoretically meaningful associative value. Defaults to FALSE.

plot_mu_all — Logical. Same as plot_mu, but produces the mu plot for every participant instead of only participant 1. Note: this compound's true value can fall below -0.5 under some conditions (see Supplementary Materials S1.8.1). When it does, the plotted value is clamped to -0.5, which is the value of a non-rewarded trial and thus the minimum theoretically meaningful associative value. Defaults to FALSE.

plot_shown — Logical. If TRUE, plots overall V by presentation number (rather than trial number) for each distinct trial-type cue combination (e.g. "A", "AB"), averaged across all participants. Phase boundaries are marked based on when that cue combination was actually chosen. Combine with plot_sd to add a shaded standard-deviation ribbon when n > 1. Defaults to FALSE.

plot_cue — Logical. If TRUE, plots the agent's V by presentation number for each individual base cue, averaged across all participants, pooling every trial type that cue appears in (e.g. "A" pools presentations of "A" and "AB" together). Note that a base cue's own value, as plotted here, is not the associative value the agent actually experiences on a given trial. This can be confusing when "AB" is itself a base cue under the default encoding: "AB" as a base cue refers only to that configural feature's own isolated value, whereas "AB" as a trial type refers to the whole compound experience, whose true value is the sum of every contributing cue ("A" + "B" + "AB"). This plot shows each base cue – including a configural cue like "AB" – as its own separate, individual line; to see the trial-level total instead, use plot_shown. Each base cue gets its own plot. Combine with plot_sd to add a shaded standard-deviation ribbon when n > 1. Defaults to FALSE.

plot_sd — Logical. If TRUE and n > 1, adds a shaded ribbon showing +/- 1 SD around the mean line on presentation-based plots (plot_cue, plot_shown, plot_appears). Has no effect with a single participant. Defaults to FALSE.

plot_raw — Logical. If TRUE, plots raw (unweighted) V for each active latent state and reward dimension, across trials, for participant 1. Belief proportion for that state is overlaid as a dashed line on a second y-axis. V is not shown until the trial after a state's creation, as that is the first trial when the latent state will actually affect the overall V. Combine with plot_label to show trial-type labels (colored by reward) on the x-axis, and plot_arm to also show the chosen arm beneath each label. Defaults to FALSE.

plot_raw_all — Logical. Same as plot_raw, but produces the raw V plot for every participant instead of only participant 1. Defaults to FALSE.

plot_q — Logical. If TRUE, plots the change-point statistic (q) across trials for participant 1, with a dashed horizontal line marking the eta threshold at which a new latent state is created. Trials where q was Inf (an immediate, unambiguous state change) are marked with a red triangle and ∞ label, capped at $2 \times$ eta for display purposes only. Combine with plot_label to show trial-type labels (colored by reward) on the x-axis, and plot_arm to also show the chosen arm beneath each label. Defaults to FALSE.

`plot_q_all` Logical. Same as `plot_q`, but produces the q plot for every participant instead of only participant 1. Defaults to `FALSE`.

`plot_sigma` Logical. If `TRUE`, plots the variance estimate (`sigma`) at the start of each trial, across trials, for participant 1. Combine with `plot_label` to show trial-type labels (colored by reward) on the x-axis, and `plot_arm` to also show the chosen arm beneath each label. Defaults to `FALSE`.

`plot_sigma_all` Logical. Same as `plot_sigma`, but produces the sigma plot for every participant instead of only participant 1. Defaults to `FALSE`.

`plot_appears` Logical. If `TRUE`, plots the total value of the chosen arm by presentation number for each base cue, averaged across all participants, whenever that cue was part of the chosen arm (alone or combined with others). Unlike `plot_cue`, which tracks a cue's own isolated value, this reflects the combined value of the whole arm on every presentation that cue contributed to. In other words, this plots the overall V of trials where a certain cue "appears". Combine with `plot_sd` to add a shaded standard-deviation ribbon when `n > 1`. Note: this compound's true value can fall below -0.5 under some conditions (see Supplementary Materials S1.8.1). When it does, the plotted value is clamped to -0.5, which is the value of a non-rewarded trial and thus the minimum theoretically meaningful associative value. Defaults to `FALSE`.

`fixed_axis` Logical. If `TRUE`, y-axes are fixed to a consistent range across plots and participants (`[-0.5, 0.5]` for V and mu plots, and a shared data-driven range for `q` and `sigma` plots), making plots directly comparable. If `FALSE`, each plot's y-axis is scaled to its own data range. Defaults to `TRUE`.

Value

A list of length `n`, one element per participant. Each element is itself a list containing:

`arm_chosen` Integer vector of length `ntrials` giving the arm chosen by the agent on each trial.

`win_received` Matrix (`ntrials` x `D`) of the reward outcome received on each trial, for each reward dimension.

`lsb` Matrix (`ntrials` x `L`) of latent-state beliefs at the start of each trial, where `ncop` is the total number of latent states.

`Lmax` Integer vector of length `ntrials` giving the number of active latent states at each trial.

`q` Numeric vector of length `ntrials` giving the change-point statistic after that trial's reward but before any new-state creation/reset (the value used for plotting).

`sigma` Numeric vector of length `ntrials` giving the agent's variance estimate at the start of each trial.

`mu` Array (`ntrials` x `D` x `A`) of the overall expected value `V` of each arm, for each reward dimension, before the agent's choice on each trial.

`mu_raw` Array (`ntrials` x `D` x `L` x `A`) of the unweighted expected value of each arm, shown separately for each latent state, for each reward dimension.

`V` Named list with one entry per base cue, each a matrix (`ntrials` x `D`) of that cue's overall associative strength over trials.

`V_cue` Named list with one entry per base cue, each a matrix (presentations x D) of that cue's own overall value, recorded only on trials where the cue was shown.

`V_raw` Named list with one entry per base cue, each an array (`ntrials` x `D` x `L`) of that cue's raw (unweighted) associative strength in every latent state over trials.

`V_shown` Named list with one entry per trial type (e.g. `"A"`, `"AB"`), each a matrix (presentations x D) of the agent's total V whenever that exact trial type was the chosen arm.

`V_appear` Named list with one entry per base cue, each a matrix (presentations x D) of the agent's total V on the chosen arm, recorded whenever that cue was part of the chosen arm.

`B` Array (`ntrials` x `C` x `C` x `D` x `L`) giving the effort matrix, which governs cue associability, for every cue pair, reward dimension, and latent state, at every trial.

`par` The full parameter and task configuration used to generate this participant. The most useful pieces are `ls` (the exact parameters used for this participant, e.g. `alpha0`, `eta`, `delta`) and `tau` (the inverse-temperature parameter used). The remaining fields (`D`, `C`, `A`, `ntrials`, `base_cues`, `ITI`, etc.) describe the task's structure and are mainly used internally (e.g. by `plot_compare()`) to recover cue counts, trial counts, and timing.

`world` List of the two arrays defining what was presented each trial: `c_vec` (`D` x `C` x `L` x `A` x `ntrials`), which cues were active per dimension, latent state, and arm; and `win` (`D` x `A` x `ntrials`), the reward generated for each arm and dimension, regardless of the agent's choice. Also includes `type_seq`, which is a list showing the order of cues presented to this participant.

`long` List (one entry per phase) of data frames combining every cue type presented in that phase, in long format, for `D = 1` only, intended for running an ANOVA using the ezANOVA package. Columns are: `subject`, the participant ID (`i + prev`, offset so IDs don't collide across combined `run()` calls); `group`, the batch label from the `group` argument; `cue_type`, the trial type or arm segment (e.g. `"A"`, `"AB"`) this row belongs to; `presentation`, the ordinal position of this row among all presentations of that cue type within this phase; `V`, the agent's overall associative value on that presentation; and, when `block` is specified, `block`, the block number that presentation falls into.

References

Cochran, A. L., & Cisler, J. M. (2019). A flexible and generalizable model of online latent-state learning. PLOS Computational Biology, 15(9). doi:10.1371/journal.pcbi.1007331

Examples

```
phase_def <- list(
  phase1 = list(
    trials = list(
      A_B  = 20,
      B_A  = 10,
      AB_C = 12,
      C_AB = 6,
      B_0  = 5
    ),
    rewards = list(
      D1 = c(A = 0.80, B = 0.20, AB = 0.50),
      D2 = c(A = 0.60, B = 0.00, AB = 0.00)
    )
  ),
  phase2 = list(
```

```
    trials = list(
      A_B  = 15,
      B_A  = 15,
      AB_C = 8,
      C_AB = 8
    ),
    rewards = list(
      D1 = c(A = 0.10, B = 0.50, C = 0.00, AB = 0.25),
      D2 = c(A = 0.00, B = 0.60, AB = 0.00)
    )
  )
)
result <- run(phase_def, n = 5)
```

Index