Package: AccumulatR (via r-universe)

October 23, 2026

Type Package

Title Simulation and Likelihood Tools for Evidence-Accumulation Models

Version 1.0.0

Description Toolbox for designing, simulating and fitting custom racing evidence-accumulation models to behavioral data. Users define a model in terms of accumulators, pooled channels, logical response rules, and timing dependencies. This allows more direct embedding of psychological hypotheses into the model. With that model you can simulate decision-making data and evaluate the likelihood of observed responses under alternative parameter values. The user-facing interface stays in R, while the simulation and likelihood calculations are handled in C++ for speed.

License GPL-3

URL https://github.com/niekstevenson/AccumulatR, https://niekstevenson.github.io/AccumulatR/

BugReports https://github.com/niekstevenson/AccumulatR/issues

Encoding UTF-8

Depends R (>= 4.1)

Imports Rcpp (>= 1.1.2), stats, utils

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

VignetteBuilder knitr

LinkingTo Rcpp (>= 1.1.2), BH

SystemRequirements C++17

NeedsCompilation yes

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

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

Date/Publication 2026-10-14 12:49:46 UTC

RemoteUrl https://github.com/niekstevenson/accumulatr

RemoteRef HEAD

RemoteSha 9646446604e2cebe3fee6fe60dc68d033fe7b053

Contents

AccumulatR-package . . . 2
add_accumulator . . . 3
add_component . . . 4
add_outcome . . . 5
add_pool . . . 6
add_trigger . . . 6
after . . . 7
all_of . . . 8
build_outcome_expr . . . 8
build_param_matrix . . . 9
complexity_metrics . . . 10
finalize_model . . . 10
first_of . . . 11
inhibit . . . 11
log_likelihood . . . 12
make_context . . . 13
none_of . . . 14
par_names . . . 14
plot_accumulators . . . 15
prepare_data . . . 16
processing_tree . . . 18
race_spec . . . 19
response_probabilities . . . 19
set_mixture . . . 20
set_parameters . . . 21
simulate . . . 22

Index **24**

`AccumulatR-package` *AccumulatR: Simulate and fit evidence-accumulation models*

Description

Details

AccumulatR is for researchers who work with response-time models in which behavior is generated by competing or interacting accumulators. The package lets you describe a model in psychologically meaningful terms, simulate behavioral data, and evaluate likelihoods for fitting.

The main workflow is:

1. Start with race_spec().
2. Add accumulators, pools, outcomes, and timing relations.
3. Call finalize_model() and supply values with build_param_matrix().
4. Generate observations with simulate(), or use experimental data.
5. Call prepare_data() and make_context(), then evaluate candidate parameters with log_likelihood().

See vignette("getting_started", package = "AccumulatR") for a worked example.

Author(s)

Maintainer: Niek Stevenson <niek.stevenson@gmail.com>

See Also

Useful links:

- https://github.com/niekstevenson/AccumulatR
- https://niekstevenson.github.io/AccumulatR/
- Report bugs at https://github.com/niekstevenson/AccumulatR/issues

---

add_accumulator *Add an accumulator to a model*

Description

An accumulator has a random finishing time drawn from dist, shifted by its onset and nondecision time t0. Use add_outcome() to connect its completion to an observed response.

Usage

```
add_accumulator(spec, id, dist, onset = 0)
```

Arguments

spec	A race_spec object.
id	Label for the accumulator.
dist	Distribution family: "lognormal", "gamma", "exgauss", "LBA", or "RDM". Names are case-insensitive.
onset	Start time for the accumulator. This can be a fixed numeric onset or a chained onset created with after().

Value

The updated `race_spec`.

See Also

`set_parameters()`, `build_param_matrix()`, `after()`

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
```

---

add_component *Define a mixture component*

Description

Components are useful when trials can come from qualitatively different processing modes, such as fast versus slow processing. Component declarations define membership only; component probabilities are configured with `set_mixture()`.

Usage

```
add_component(spec, id, members, n_outcomes = NULL)
```

Arguments

`spec`	A `race_spec` object.
`id`	Component label.
`members`	Accumulator labels that belong to this component.
`n_outcomes`	Optional component-specific override for the number of observed ordered responses.

Value

The updated `race_spec`.

add_outcome — *Define an observed response*

Description

Attach a response label to a finishing rule. In a single-response model, the first available outcome determines the observed response and time.

Usage

```
add_outcome(spec, label, expr, options = list())
```

Arguments

`spec`	A `race_spec` object.
`label`	Response label that should appear in the behavioral data.
`expr`	Accumulator or pool label, or a rule built with `first_of()`, `all_of()`, `none_of()`, `inhibit()`, or `build_outcome_expr()`.
`options`	Named list of response settings: • `component`: component labels in which this outcome is active. Omit to use it in every component. • `map_outcome_to`: another declared outcome label, or `NA_character_` to record no response when this outcome wins. • `guess`: a list with declared outcome `labels`, their probability `weights` (summing to one), and `rt_policy = "keep"` or `"na"`. When this outcome wins, draw its recorded label using these weights; `"na"` discards the response time. The default policy is `"keep"`. Guessing and remapping require `n_outcomes = 1`.

Value

The updated `race_spec`.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
```

add_pool — *Pool several accumulators under a shared label*

Description

A pool finishes when its kth member finishes. Use the pool label in `add_outcome()`, another pool, or `after()`.

Usage

```
add_pool(spec, id, members, k = 1L)
```

Arguments

`spec`	A `race_spec` object.
`id`	Label for the pool.
`members`	Accumulator or pool labels included in the pool.
`k`	Number of members that must finish, from `1` to `length(members)`. The default `1` gives the first member's finishing time.

Value

The updated `race_spec`.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_accumulator(spec, "B", "lognormal")
spec <- add_pool(spec, "P1", members = c("A", "B"), k = 1L)
```

add_trigger — *Add a shared absence trigger*

Description

With probability given by `name`, all member accumulators are absent on a trial. Otherwise they follow their specified onsets and finishing-time distributions. Use separate triggers for independent absence events; `set_parameters()` can give those events a common probability.

Usage

```
add_trigger(spec, name, members)
```

Arguments

spec	A race_spec object.
name	Trigger parameter name. Supply its probability in [0, 1] to build_param_matrix().
members	Accumulator labels controlled by the shared absence draw.

Value

The updated race_spec.

---

after — *Start one accumulator after another process finishes*

Description

This is useful for staged or contingent architectures, where one process can only begin after an earlier accumulator or pool has finished.

Usage

```
after(source, lag = 0)
```

Arguments

source	Accumulator or pool label that must finish first.
lag	Optional non-negative delay added after source finishes.

Value

A chained-onset specification for add_accumulator(onset = ...).

Examples

```
after("A")
after("pool1", lag = 0.05)
```

all_of — *Define a response that requires several processes to finish*

Description

Return a rule that finishes when all required events have completed. Any `none_of()` conditions are checked at that finishing time.

Usage

```
all_of(...)
```

Arguments

`...`	Accumulator labels, pool labels, or response expressions.

Value

An expression object.

Examples

```
all_of("A", "B")
```

build_outcome_expr — *Build a response rule from a quoted expression*

Description

Convert a source label or quoted R expression into a response rule for `add_outcome()`. Within a quoted expression, `&` combines requirements, `|` allows alternative routes, and `!` specifies an absence condition. These correspond to `all_of()`, `first_of()`, and `none_of()`.

Usage

```
build_outcome_expr(expr)
```

Arguments

`expr`	Accumulator or pool label, symbol, quoted logical expression, or an expression object returned by an outcome helper.

Value

An expression object used inside model specifications.

Examples

```
build_outcome_expr(quote(A & !B))
```

---

build_param_matrix *Create trial-level parameter values*

Description

Expand a named parameter vector into the matrix used by `simulate()` and `log_likelihood()`. Each trial receives the same parameter values; rows are grouped by trial, with accumulators in their model declaration order.

Usage

```
build_param_matrix(model, param_values, n_trials = 1L)
```

Arguments

`model`	Finalized model structure.
`param_values`	Named numeric vector using the names from `par_names()`. Omitted nondecision-time parameters (`t0`) default to zero. Supply all other parameters on their natural scale, after any fitting transformations.
`n_trials`	Positive integer number of trial blocks to create.

Details

Parameter domains are checked before expansion. All values must be finite, `t0` must be nonnegative, and distribution scales must be positive. Trigger probabilities must lie in `[0, 1]`; sampled mixture weights must be nonnegative and sum to at most one. See `vignette("distributions", package = "AccumulatR")` for distribution-specific constraints.

For trial-specific parameters, modify the relevant matrix rows while preserving their order and column layout. Likelihood and simulation calls use those values directly, so modified values must satisfy the same domains.

Value

A numeric matrix with one row per trial/accumulator pair. Columns contain trigger absence probability `q`, nondecision time `t0`, distribution parameters `p1`, `p2`, and so on, followed by sampled mixture weights. Distribution slots follow the order listed in the Supported Distributions vignette. Accumulators without a trigger have `q = 0`.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
vals <- c(m = 0, s = 0.1)
build_param_matrix(finalize_model(spec), vals, n_trials = 2)
```

complexity_metrics *Inspect the size of a compiled likelihood plan*

Description

Report how many symbolic regions, numerical operations, and integration kernels the model requires. Use these counts to investigate models whose context construction or likelihood evaluation is expensive.

Usage

```
complexity_metrics(context)
```

Arguments

`context`	Context created with `make_context(diagnostics = TRUE)`.

Value

A list containing a `variants` data frame and a `total` list. Each variant is a compiled component plan. Columns count symbolic regions and cells, compiled roots and nodes, and integral kernels. Fields beginning with `max_` report maxima; other total fields sum across variants.

finalize_model *Finalize a model for simulation and fitting*

Description

Check source references, timing dependencies, response rules, and parameter grouping, and create the model object used by `simulate()`, `prepare_data()`, and `build_param_matrix()`. Build its likelihood context with `make_context()`.

Usage

```
finalize_model(model)
```

Arguments

`model`	Model specification.

Value

A `model_structure` object.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
finalize_model(spec)
```

---

first_of — *Define a response that occurs when the first listed process finishes*

Description

Return a rule whose finishing time is the earliest completion among its arguments. Each argument must contain an event that can generate a response.

Usage

```
first_of(...)
```

Arguments

`...`	Accumulator labels, pool labels, or response expressions.

Value

An expression object.

Examples

```
first_of("A", "B")
```

---

inhibit — *Define a response that is blocked by another process*

Description

The response occurs when `reference` finishes, provided `by` has not finished strictly earlier. A blocker that finishes later does not cancel a response that has already occurred.

Usage

```
inhibit(reference, by)
```

Arguments

`reference`	Response rule, accumulator label, or pool label to be blocked.
`by`	Blocking process or expression.

Value

A guarded expression object.

Examples

```
inhibit("A", "B")
```

---

log_likelihood *Evaluate log-likelihoods of behavioral data*

Description

Compute the summed log-likelihood by default, or trial-wise log-likelihoods when `sum = FALSE`. Build the context and prepare the observations once, then reuse them when evaluating candidate parameter matrices for the same model.

Usage

```
log_likelihood(
  context,
  data,
  parameters,
  ok = NULL,
  sum = TRUE,
  min_ll = log(1e-10)
)
```

Arguments

`context`	Context created with `make_context()`.
`data`	Prepared data created with `prepare_data()`.
`parameters`	Numeric matrix from `build_param_matrix()`, with one accumulator block per prepared trial in matching order.
`ok`	Optional logical vector with one value per prepared trial. `TRUE` evaluates that trial; `FALSE` assigns `min_ll`. These assigned values are included in the sum. For compressed data, use the retained trial order.
`sum`	If `TRUE`, return the summed log-likelihood. If `FALSE`, return trial-wise log-likelihood values.
`min_ll`	Minimum log-likelihood value used for excluded or impossible trials.

Details

Response-time observations contribute densities, so a log-likelihood can be positive. Missing responses and censoring contribute probability masses according to the observation rules.

Use `prepare_data()` and `build_param_matrix()` for the same model. Evaluation assumes matching layouts and valid parameters and does not repeat preparation checks.

Value

A summed log-likelihood by default, or a numeric vector of trial-wise log-likelihood values when `sum = FALSE`.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
structure <- finalize_model(spec)
params_df <- build_param_matrix(
  structure,
  c(m = 0, s = 0.1),
  n_trials = 2
)
data_df <- simulate(structure, params_df, seed = 1)
prepared <- prepare_data(structure, data_df)
ctx <- make_context(structure)
log_likelihood(ctx, prepared, params_df)
```

`make_context` *Build a compiled likelihood context from a model*

Description

Compile the model's response rules and dependencies for likelihood evaluation. Reuse the context across candidate parameter values and datasets for the same model. Prepare each dataset with `prepare_data()`.

Usage

```
make_context(structure, diagnostics = FALSE)
```

Arguments

`structure`	Finalized model structure.
`diagnostics`	If `TRUE`, collect model compilation statistics for `complexity_metrics()`.

Value

An `accumulatr_context` object.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
structure <- finalize_model(spec)
make_context(structure)
```

none_of — *Define the absence of an event*

Description

Use inside `all_of()` to require that `expr` has not finished before the conjunction completes. Absence supplies a condition, not a response time, so it cannot be an outcome on its own or a standalone `first_of()` branch.

Usage

```
none_of(expr)
```

Arguments

`expr`	Accumulator label, pool label, or expression to negate.

Value

An expression object.

Examples

```
all_of("go", none_of("stop"))
```

par_names — *List the parameter names used by a model*

Description

Return the names accepted by `build_param_matrix()`, after applying any grouping or renaming in `set_parameters()`. The list includes nondecision times, trigger probabilities, and sampled mixture weights where applicable. These names do not determine which parameters an optimizer must estimate; parameters can be held fixed by supplying constant values.

Usage

```
par_names(model)
```

Arguments

`model`	A `race_spec` or finalized `model_structure` object.

Value

A character vector of parameter names.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
par_names(spec)
```

---

plot_accumulators *Plot accumulators and their timing relations*

Description

Draw a schematic view of the model with time on the x-axis and evidence on the y-axis. Each accumulator is shown as a simple trajectory, and timing or blocking relations are overlaid so you can inspect the qualitative structure of the model before fitting.

Usage

```
plot_accumulators(
  model,
  xlim = NULL,
  ylim = c(0, 1),
  line_length = 1,
  angle_dodge = 10,
  accumulator_order = NULL,
  curve_strength = 0.18,
  line_col = "gray20",
  line_lwd = 3,
  axis_lwd = 3,
  blocker_lwd = 2.6,
  show_labels = TRUE,
  main = "Accumulator Schematic",
  xlab = "Time",
  ylab = "Evidence",
  ...
)
```

Arguments

`model`	A finalized model structure.
`xlim`	Optional x-axis limits. If `NULL`, they are chosen from the model.
`ylim`	Y-axis limits.
`line_length`	Length of each accumulator trajectory in plot units.
`angle_dodge`	Target angular spacing in degrees between neighbouring accumulators. Angles are centered at 45 degrees and constrained to 10 to 80 degrees. If there are too many accumulators sharing the same onset, spacing is reduced uniformly within that onset group.

accumulator_order	Optional character vector of accumulator labels, ordered from top-to-bottom. This only affects accumulators sharing the same onset.
curve_strength	Curvature applied to relationship arcs.
line_col	Color for accumulator trajectories.
line_lwd	Line width for accumulator trajectories.
axis_lwd	Line width for x/y axis arrows.
blocker_lwd	Line width for blocker arrows.
show_labels	If TRUE, add accumulator labels near the line endpoints.
main	Main title.
xlab	X-axis label.
ylab	Y-axis label.
...	Additional arguments passed to graphics::plot.

Value

Invisibly returns a list with plotting coordinates and inferred structural links.

Examples

```
spec <- race_spec() |>
  add_accumulator("go", "lognormal") |>
  add_accumulator("stop", "lognormal", onset = 0.15) |>
  add_pool("P", c("go", "stop")) |>
  add_outcome("R", "P") |>
  finalize_model()
plot_accumulators(spec)
```

---

prepare_data *Prepare behavioral data for likelihood evaluation*

Description

Check response labels, times, and observation conditions, and arrange the data into one row per accumulator per trial for log_likelihood().

Usage

```
prepare_data(structure, data_df, compress = FALSE)
```

Arguments

`structure`	Finalized model structure.
`data_df`	Data frame with response labels `R` and numeric response times `rt`, usually one row per trial. Labels must match the model's outcomes. Optional columns specify `component`, `onset`, ranked responses (`R2`, `rt2`, and so on), or censoring and truncation; see Details.
`compress`	If `TRUE`, collapse repeated prepared trials and attach an `expand` index mapping original trials to retained trials. Use only when repeated observations also share parameter values. Supply parameters and `ok` for the retained trials; `log_likelihood()` expands the returned values to the original trial order.

Details

Trial layout. Without a `racer` column, each row is one trial. Trials are numbered consecutively in input order. To supply accumulator-specific onsets, include `trials` and `racer` columns with one complete accumulator block per trial in model order. Repeat each trial's response and component values across its block. An `onset` replaces a fixed onset or adds an offset to a chained onset. Omit it to use model defaults.

Mixtures. A nonmissing `component` label conditions on that component. An omitted or `NA` label averages over the model's mixture probabilities.

Missing observations. For single-response models, `R = NA` and `rt = NA` denote no observed response. A finite `rt` requires a response label. A known response with `rt = NA` requires a censoring code or a model with an observation rule such as `guess` or `map_outcome_to`.

Ranked responses. Supply paired columns `R2`/`rt2`, `R3`/`rt3`, and so on. The first pair must be observed. Labels cannot repeat, and observed times must increase strictly. Later pairs may both be `NA`; all subsequent ranks must then also be missing. Ranked observations support neither censoring/truncation nor guessing/remapping rules.

Censoring and truncation. `LT` and `UT` define the observation window. An active truncation window conditions the likelihood on an observable response with $LT \leq rt \leq UT$. For a censored trial set `rt = NA` and use:

- `missingness = 1` for $LT \leq rt < LC$;
- `missingness = 2` for $UC < rt \leq UT$;
- `missingness = 3` for the union of those intervals.

`R` may retain a known response or be `NA`. Uncensored trials use `missingness = NA`. Missing bounds default to `LT = LC = 0` and `UC = UT = Inf`. These bounds describe the recorded response time. Times exactly at `LC` or `UC` remain uncensored.

Value

A data frame of class `accumulatr_data`, with accumulator rows grouped by trial and ordered as in the model. Response and component labels are factors with model-defined levels; attributes store the likelihood layout.

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
structure <- finalize_model(spec)
params_df <- build_param_matrix(
  structure,
  c(m = 0, s = 0.1),
  n_trials = 2
)
data_df <- simulate(structure, params_df, seed = 1)
prepare_data(structure, data_df)
```

---

processing_tree *Draw a processing tree for the responses in a model*

Description

Show the observed responses, the accumulators or pools that feed them, and any blocking relationships as a directed graph.

Usage

```
processing_tree(model, outcome_label = NULL, return_dot = FALSE)
```

Arguments

model	A finalized model structure.
outcome_label	Optional response label. If supplied, only that response is shown.
return_dot	If TRUE, return the graph description as a list instead of rendering it with DiagrammeR.

Value

If DiagrammeR is available and return_dot = FALSE, a DiagrammeR graph. Otherwise, a list with dot, nodes, and edges.

race_spec — *Start a race-model specification*

Description

Create an empty specification. Add accumulators and response rules, then call `finalize_model()` before simulation or likelihood evaluation.

Usage

```
race_spec(n_outcomes = 1L)
```

Arguments

`n_outcomes`	Number of ordered observed responses to retain per trial. Use `1` for standard choice/RT data, `2` when you also observe the second finishing response, and so on.

Value

A `race_spec` object.

Examples

```
race_spec()
```

response_probabilities — *Evaluate marginal response probabilities*

Description

Calculate the probability of each first response, integrated over response time and averaged over mixture components. With multiple trial blocks in `parameters`, return the mean probabilities across those blocks.

Usage

```
response_probabilities(context, parameters, include_na = TRUE)
```

Arguments

`context`	Context created with `make_context()`.
`parameters`	Parameter matrix from `build_param_matrix()`. Use `n_trials = 1` for one set of response probabilities.
`include_na`	If `TRUE`, include a `"NA"` entry when there is residual probability of no observed response.

Value

A named numeric vector of marginal response probabilities. Names are observed outcome labels. When `include_na = TRUE`, a residual `"NA"` entry is included if the model assigns probability mass to unobserved or `NA`-mapped outcomes.

Examples

```
spec <- race_spec() |>
  add_accumulator("left", "lognormal") |>
  add_accumulator("right", "lognormal") |>
  add_outcome("left", "left") |>
  add_outcome("right", "right") |>
  set_parameters(separate = list(m = TRUE))

model <- finalize_model(spec)
params <- build_param_matrix(
  model,
  c(left.m = log(0.25), right.m = log(0.40), s = 0.20),
  n_trials = 1
)

response_probabilities(make_context(model), params)
```

---

set_mixture *Control how mixture components are combined*

Description

Fixed mixtures use known component probabilities. Sampled mixtures expose automatic `p.<component>` parameters for every non-reference component; the reference component receives the residual probability. Both modes draw a component during simulation. In likelihood evaluation, a supplied component label conditions on that component; an absent or `NA` label averages over components using their probabilities.

Usage

```
set_mixture(
  spec,
  mode = c("fixed", "sample"),
  weights = NULL,
  reference = NULL
)
```

Arguments

`spec`	A `race_spec` object.
`mode`	`"fixed"` stores probabilities in the model. `"sample"` makes probabilities parameters supplied to `build_param_matrix()`.

weights	Named numeric component probabilities for fixed mixtures. If NULL, fixed mixtures use uniform component probabilities.
reference	Reference component for sampled mixtures. Its probability is one minus the sum of non-reference probabilities. Defaults to the last component added to the model.

Value

The updated race_spec.

---

set_parameters *Control parameter grouping and names*

Description

Parameters are grouped by compatible type by default. For example, two lognormal accumulators expose one m, one s, and one t0 parameter unless you ask for specific parameters to be separate. Triggers expose their trigger name directly, and sampled mixtures expose automatic p.<component> parameters.

Usage

```
set_parameters(spec, separate = NULL, share = NULL, rename = NULL)
```

Arguments

spec	A race_spec object.
separate	Named list. Each name is a grouped public parameter, and each value is one or more accumulator ids to split from that group. Use TRUE to split every member of a group.
share	Named list mapping a new public name to default public names or internal parameter names that should share one value.
rename	Named character vector mapping current public names to new public names.

Value

The updated race_spec.

Examples

```
spec <- race_spec() |>
  add_accumulator("go", "lognormal") |>
  add_accumulator("stop", "lognormal") |>
  add_outcome("go", "go") |>
  add_outcome("stop", "stop") |>
  set_parameters(
    separate = list(m = c("go", "stop")),
```

```
    rename = c(s = "spread", t0 = "nondecision")
  )

par_names(spec)
```

---

simulate *Simulate behavioral data from a model*

Description

Generate one trial for each accumulator block in `params_df`, using the model's response rules, timing dependencies, triggers, and mixture weights.

Usage

```
simulate(
  structure,
  params_df,
  trial_df = NULL,
  seed = NULL,
  keep_detail = FALSE,
  keep_component = NULL
)
```

Arguments

`structure`	Finalized model structure.
`params_df`	Parameter matrix from `build_param_matrix()`, with one row per accumulator per trial, in the order accumulators were added to the model.
`trial_df`	Optional trial-by-accumulator conditioning table containing complete `trials`/`racer` blocks in model order. It may supply per-trial `component` and per-accumulator `onset` values. Repeat a component label across all rows of its trial; `NA` draws a component from the mixture. An onset value replaces a fixed onset or adds a delay to a chained onset; `NA` uses the onset specified in the model.
`seed`	Optional seed passed to `set.seed()`. If `NULL`, use the current state of R's random-number generator.
`keep_detail`	If `TRUE`, attach a `details` list containing latent source times and outcome candidates for each trial. Inactive or unreachable sources have infinite completion times.
`keep_component`	Whether to keep the chosen mixture component in the output when the model has multiple components. If `NULL`, fixed mixtures keep the component label and sampled mixtures drop it.

Details

Use the same model to build `params_df` and simulate the data. Supply valid parameter values and keep the matrix and conditioning table in matching row order; simulation does not check their layout or domains.

Value

A data frame with one row per trial and columns `trials`, `R`, and `rt`. A trial with no observed response has `R = NA` and `rt = NA`. If `n_outcomes > 1`, additional ordered response pairs such as `R2`/`rt2` are included; unobserved later ranks are `NA`.

See Also

`prepare_data()`, `log_likelihood()`, `set_mixture()`

Examples

```
spec <- race_spec()
spec <- add_accumulator(spec, "A", "lognormal")
spec <- add_outcome(spec, "A_win", "A")
structure <- finalize_model(spec)
params <- c(m = 0, s = 0.1)
df <- build_param_matrix(structure, params, n_trials = 3)
simulate(structure, df, seed = 123)
```

Index

∗ **package**
AccumulatR-package, 2

AccumulatR *(AccumulatR-package)*, 2
AccumulatR-package, 2
add_accumulator, 3
add_component, 4
add_outcome, 5
add_outcome(), *3*, *6*, *8*
add_pool, 6
add_trigger, 6
after, 7
after(), *4*, *6*
all_of, 8
all_of(), *5*, *8*, *14*

build_outcome_expr, 8
build_outcome_expr(), *5*
build_param_matrix, 9
build_param_matrix(), *3*, *4*, *7*, *10*, *12*, *14*, *19*, *20*, *22*

complexity_metrics, 10
complexity_metrics(), *13*

finalize_model, 10
finalize_model(), *3*, *19*
first_of, 11
first_of(), *5*, *8*, *14*

inhibit, 11
inhibit(), *5*

log_likelihood, 12
log_likelihood(), *3*, *9*, *16*, *17*, *23*

make_context, 13
make_context(), *3*, *10*

none_of, 14
none_of(), *5*, *8*

par_names, 14
par_names(), *9*
plot_accumulators, 15
prepare_data, 16
prepare_data(), *3*, *10*, *12*, *13*, *23*
processing_tree, 18

race_spec, 19
race_spec(), *3*
response_probabilities, 19

set.seed(), *22*
set_mixture, 20
set_mixture(), *23*
set_parameters, 21
set_parameters(), *4*, *6*, *14*
simulate, 22
simulate(), *3*, *9*, *10*