Package 'meow'

July 6, 2026

Title Unified Framework for Computer Adaptive Testing Simulations

Version 1.0.0

Description Provides an extensible framework for conducting simulations to compare data generating processes, item selection algorithms, parameter update algorithms, and stopping rules in computer adaptive testing (CAT) applications. Bundled algorithms include the Elo-based update rules of Klinkenberg, Straatemeier and van der Maas (2011) <doi:10.1016/j.compedu.2011.02.003> and Vermeiren, Kruis, Bolsinova, van der Maas and Hofman (2025) <doi:10.1016/j.caeai.2025.100376>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports Rfast, stats

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

URL https://klintkanopka.com/meow/

BugReports https://github.com/klintkanopka/meow/issues

VignetteBuilder knitr

Config/testthat/edition 3

Depends R (>= 3.5)

NeedsCompilation no

Author Klint Kanopka [aut, cre] (ORCID: <https://orcid.org/0000-0003-3196-9538>), Sophia Deng [aut]

Maintainer Klint Kanopka <klint.kanopka@nyu.edu>

Repository CRAN

Date/Publication 2026-07-06 13:30:08 UTC

Contents

`construct_adj_mat` *Construct an item-pool adjacency matrix.*

Description

For an item pool with N items, this returns an N x N matrix. The diagonal elements contain the number of times each item has been administered. The off-diagonal element (i, j) contains the number of respondents who have been administered both item i and item j. In general this function is not called directly, but is instead called within `meow()`. It is exposed to aid users who are testing item selection functions they have written.

Usage

```
construct_adj_mat(admin)
```

Arguments

`admin`	An administration matrix with one row per respondent and one column per item. Non-zero entries indicate that an item has been administered to a respondent (see `meow()` for details of the matrix-based simulation state). A logical matrix is also accepted.

Value

An item-item adjacency matrix of type `matrix`.

Examples

```
admin <- matrix(c(1, 1, 0,
                  1, 0, 1), nrow = 2, byrow = TRUE)
construct_adj_mat(admin)
```

`data_existing` *Load data from existing files*

Description

`data_existing()` is a wrapper for three separate calls to `read.csv()` that packages the output into the object used by `meow()`.

Usage

```
data_existing(resp_path, pers_path, item_path)
```

Arguments

`resp_path`	A file path to a long form .csv file. File should have three columns, `id` which contains a numeric respondent identifier, `item` which contains a numeric item identifier, and resp which contains an item response. Be sure the form of the item response comports with the parameter update functions you choose to use.
`pers_path`	A file path to a wide form .csv file that contains true person parameter values, with one person per row. Include a person index column, named `id`. Default column name for unidimensional person ability should be `theta`
`item_path`	A file path to a wide form .csv file that contains true item parameter values, with one item per row. Include an item index column, named `item`. Default column names for difficulty should be `b` and default column name for discrimination should be `a`,

Value

A list with three components: A dataframe of item response named `resp`, a dataframe of true person parameters named `pers_tru`, and a dataframe of true item parameters named `item_tru`

data_simple_1pl *A default data generation function that simulates normally distributed respondent abilities and item difficulties*

Description

`data_simple_1pl()` constructs data according to a simple one parameter logistic IRT model. The user may specify a number of persons, a number of items, and a random seed for reproducibility. Person abilities and item difficulties are both drawn from a standard normal.

Usage

```
data_simple_1pl(N_persons = 100, N_items = 50, data_seed = 242424)
```

Arguments

`N_persons`	Number of respondents to simulate
`N_items`	Number of items to simulate
`data_seed`	A random seed for generating reproducible data. This seed is re-initialized at the end of the data generation process

Value

A list with three components: A dataframe of item response named `resp`, a dataframe of true person parameters named `pers_tru`, and a dataframe of true item parameters named `item_tru`

Examples

```
data <- data_simple_1pl(N_persons = 10, N_items = 8)
str(data)
```

edge_weight_inverse *Alternative edge weight functions for network-based item selection*

Description

These functions provide different approaches to calculating edge weights from the adjacency matrix.

Usage

```
edge_weight_inverse(adj_mat, alpha = 1)

edge_weight_negative_log(adj_mat, alpha = 1)

edge_weight_linear(adj_mat, max_co_responses = NULL)

edge_weight_power(adj_mat, beta = 0.5, alpha = 1)

edge_weight_exponential(adj_mat, lambda = 0.1, alpha = 1)
```

Arguments

`adj_mat`	The adjacency matrix where entry i,j is the number of co-responses between items i and j
`alpha`	Smoothing parameter for avoiding division by zero
`max_co_responses`	Scaling factor for linear weighting
`beta`	Exponent for power transformation
`lambda`	Decay constant for exponential decay weighting

Value

A matrix of edge weights for use in distance calculations

Examples

```
adj_mat <- matrix(c(3, 1, 1, 2), nrow = 2)
edge_weight_inverse(adj_mat)
```

`meow` *Conduct a full CAT simulation.*

Description

`meow()` is the core function of this simulation framework. It exists to help users compare efficiency tradeoffs across different item selection algorithms, parameter update algorithms, and data generating processes. It takes as arguments an item selection function, a parameter update function, and a data loader function and uses these to carry out a simulation of a full CAT administration. Default behavior is to proceed until no further items are administered. Because the internal simulation logic stops as soon as an iteration administers no new items, early stopping conditions should be implemented within the item selection function (by declining to administer further items).

Usage

```
meow(
  select_fun,
  update_fun,
  data_loader,
  select_args = list(),
  update_args = list(),
  data_args = list(),
  init = NULL,
  fix = "none",
  keep_adj_mats = TRUE
)
```

Arguments

`select_fun`	A function that specifies the item selection algorithm.
`update_fun`	A function that specifies the parameter update algorithm.
`data_loader`	A function that specifies the data generating process.
`select_args`	A named list of arguments to be passed to `select_fun`.
`update_args`	A named list of arguments to be passed to `update_fun`.
`data_args`	A named list of arguments to be passed to `data_loader`.
`init`	A list of initialization values for estimated person and item parameters. Accepts a named list with two entries, `pers` and `item`, giving the initial estimated parameter data frames. Defaults to `NULL`, which initializes all estimated parameters to zero.
`fix`	Which estimated parameters to treat as fixed at their true values. One of `none` (the default), `pers`, `item`, or `both`.
`keep_adj_mats`	Logical; if `TRUE` (the default) an adjacency matrix is stored for every iteration. If `FALSE`, only the final adjacency matrix is retained, which saves memory for large item pools or long simulations.

Details

Simulation state:

For speed, `meow()` represents responses with matrices rather than long data frames. Two matrices, each with one row per respondent and one column per item, are passed to the user-supplied modules:

- `R` — the (potential) response of every respondent to every item. This is produced once from the long `resp` data frame returned by the data loader.
- `admin` — an integer administration matrix. An entry of `0` means the item has not been administered to that respondent; a positive entry means it has, and the value encodes the order of administration. Use `admin != 0` (or `meow_administered()`) as an administered mask.

Person and item *parameters* are kept as data frames (`pers` and `item`), each with an identifier column (`id` and `item`, respectively) followed by one column per parameter, so that users retain the flexibility to add arbitrary parameters.

Module contracts:

An **item selection** function receives `pers`, `item`, `R`, `admin`, and `adj_mat` (plus any `select_args`) and returns an administration matrix with newly selected cells marked non-zero. The harness stamps the order of administration, so a function need only set newly selected cells to a positive value (or `TRUE`) while leaving previously administered cells unchanged.

A **parameter update** function receives `pers`, `item`, `R`, and `admin` (plus any `update_args`) and returns a list with updated `pers` and `item` data frames.

Module authors who prefer long data frames can convert with `meow_long()`.

Value

A list of four named entities. `results` is a data frame with one row per iteration of the simulation. It contains an `iter` column for the iteration number and two columns per person and item parameter, one for the estimated parameter and one for the bias in that estimate. `adj_mats` is a list of item-item adjacency matrices, one per iteration (or, when `keep_adj_mats = FALSE`, a single-element list with the final matrix); edge weights count the number of respondents administered each pair of items. `pers_tru` and `item_tru` are the true person and item parameter data frames.

Examples

```
sim <- meow(
  select_fun = select_max_info,
  update_fun = update_theta_mle,
  data_loader = data_simple_1pl,
  data_args = list(N_persons = 20, N_items = 15),
  fix = "item"
)
head(sim$results)
```

---

meow_administered *Logical mask of administered items.*

Description

A convenience helper for use inside user-written modules. Returns a logical matrix that is `TRUE` wherever an item has been administered to a respondent.

Usage

```
meow_administered(admin)
```

Arguments

`admin`	An administration matrix (see `meow()`).

Value

A logical matrix the same shape as `admin`.

Examples

```
admin <- matrix(c(1L, 2L, 0L, 1L), nrow = 2)
meow_administered(admin)
```

`meow_long` *Convert the matrix simulation state to a long data frame of responses.*

Description

`meow()` represents responses as a respondent-by-item matrix (`R`) together with an administration matrix (`admin`). This helper returns the administered responses as a long data frame with columns `id`, `item`, and `resp`, ordered by respondent and then by the order in which items were administered. It is the recommended bridge for module authors who prefer to work with tidyverse-style long data inside their own item selection or parameter update functions.

Usage

```
meow_long(R, admin)
```

Arguments

`R`	A respondent-by-item matrix of (potential) responses.
`admin`	An administration matrix the same shape as `R`. Non-zero entries indicate administered items; positive integer entries additionally encode the order of administration.

Value

A long-form data frame with columns `id`, `item`, and `resp` containing only the administered responses.

Examples

```
R <- matrix(c(1, 0, 1, 1), nrow = 2)
admin <- matrix(c(1L, 0L, 2L, 1L), nrow = 2)
meow_long(R, admin)
```

`select_max_dist` *Item selection by network distance criterion.*

Description

Administers the item farthest in the item network from the items a respondent has already answered, with edges weighted by the inverse of their entry in the item-item adjacency matrix. Ties are broken using the maximum information criterion.

Usage

```
select_max_dist(pers, item, R, admin, adj_mat = NULL, n_candidates = 1)
```

Arguments

`pers`	A data frame of current respondent ability estimates.
`item`	A data frame of current item parameter estimates.
`R`	A respondent-by-item matrix of potential responses.
`admin`	An integer administration matrix; `0` indicates an item has not been administered to a respondent. See `meow()` for details.
`adj_mat`	An item-item adjacency matrix. See `construct_adj_mat()`.
`n_candidates`	The number of farthest items to assemble into a candidate pool before selecting the next item by maximum information. Allows users to trade off network density against estimation efficiency.

Value

An updated administration matrix with the selected item marked for each respondent.

Examples

```
sim <- meow(select_max_dist, update_theta_mle, data_simple_1pl,
            data_args = list(N_persons = 10, N_items = 10), fix = "item")
nrow(sim$results)
```

`select_max_dist_enhanced` *Network-based item selection with configurable edge weights.*

Description

Extends `select_max_dist()` with a flexible edge weight calculation.

Usage

```
select_max_dist_enhanced(
  pers,
  item,
  R,
  admin,
  adj_mat = NULL,
  n_candidates = 1,
  edge_weight_fun = edge_weight_inverse,
  edge_weight_args = list()
)
```

Arguments

`pers`	A data frame of current respondent ability estimates.
`item`	A data frame of current item parameter estimates.
`R`	A respondent-by-item matrix of potential responses.
`admin`	An integer administration matrix; `0` indicates an item has not been administered to a respondent. See `meow()` for details.
`adj_mat`	An item-item adjacency matrix. See `construct_adj_mat()`.
`n_candidates`	The number of farthest items to assemble into a candidate pool before selecting the next item by maximum information. Allows users to trade off network density against estimation efficiency.
`edge_weight_fun`	A function that computes edge weights from the adjacency matrix. See `edge_weight_inverse()`.
`edge_weight_args`	A named list of additional arguments for `edge_weight_fun`.

Value

An updated administration matrix with the selected item marked for each respondent.

Examples

```
sim <- meow(select_max_dist_enhanced, update_theta_mle, data_simple_1pl,
            data_args = list(N_persons = 10, N_items = 10), fix = "item",
            select_args = list(edge_weight_fun = edge_weight_power))
nrow(sim$results)
```

`select_max_info` *Item selection by maximum Fisher information.*

Description

Administers the remaining item with the highest information for each respondent, computed from the current parameter estimates and a 2PL item response function.

Usage

```
select_max_info(pers, item, R, admin, adj_mat = NULL)
```

Arguments

`pers`	A data frame of current respondent ability estimates.
`item`	A data frame of current item parameter estimates.
`R`	A respondent-by-item matrix of potential responses.
`admin`	An integer administration matrix; `0` indicates an item has not been administered to a respondent. See `meow()` for details.
`adj_mat`	An item-item adjacency matrix. See `construct_adj_mat()`.

Value

An updated administration matrix with the most informative remaining item marked for each respondent.

Examples

```
sim <- meow(select_max_info, update_theta_mle, data_simple_1pl,
            data_args = list(N_persons = 10, N_items = 10), fix = "item")
nrow(sim$results)
```

select_random *Item selection by random draw from the remaining item bank.*

Description

Each respondent's next item is drawn at random from the items they have not yet been administered.

Usage

```
select_random(pers, item, R, admin, adj_mat = NULL, select_seed = NULL)
```

Arguments

pers	A data frame of current respondent ability estimates.
item	A data frame of current item parameter estimates.
R	A respondent-by-item matrix of potential responses.
admin	An integer administration matrix; `0` indicates an item has not been administered to a respondent. See `meow()` for details.
adj_mat	An item-item adjacency matrix. See `construct_adj_mat()`.
select_seed	A random seed used only for item selection. The seed is cleared after use so that successive simulations vary unless a seed is given.

Value

An updated administration matrix with a random next item marked for each respondent.

Examples

```
sim <- meow(select_random, update_theta_mle, data_simple_1pl,
            data_args = list(N_persons = 10, N_items = 10), fix = "item",
            select_args = list(select_seed = 1))
nrow(sim$results)
```

select_restrict_rate *Maximum-information item selection with an exposure-rate cap.*

Description

A maximum Fisher information selector with a simple exposure control. Each item's share of all administrations so far (the diagonal of `adj_mat`, normalized to sum to one) is treated as an exposure rate, and items whose rate has reached `r_max` are withheld. The most informative permitted item is then administered to each respondent. If a respondent has no permitted unadministered item, they receive no item that iteration; when this occurs for every remaining respondent at once, the simulation administers nothing new and stops, so this selector also acts as an implicit stopping rule.

Usage

```
select_restrict_rate(pers, item, R, admin, adj_mat = NULL, r_max = 0.025)
```

Arguments

pers	A data frame of current respondent ability estimates.
item	A data frame of current item parameter estimates.
R	A respondent-by-item matrix of potential responses.
admin	An integer administration matrix; 0 indicates an item has not been administered to a respondent. See meow() for details.
adj_mat	An item-item adjacency matrix. See construct_adj_mat().
r_max	The maximum permitted exposure rate (an item's share of all administrations) before that item is withheld. Defaults to 0.025.

Details

Because the exposure rate is each item's share of all administrations, its average across items is 1 / N_items. Values of r_max above 1 / N_items rarely bind, values near it bind only transiently, and values below it induce early stopping.

Value

An updated administration matrix with the most informative permitted item marked for each respondent who still has one.

Examples

```
sim <- meow(select_restrict_rate, update_theta_mle, data_simple_1pl,
            data_args = list(N_persons = 10, N_items = 10), fix = "item",
            select_args = list(r_max = 0.2))
nrow(sim$results)
```

---

select_sequential *Item selection by item id, simulating a fixed test form.*

Description

This function administers the next unadministered item to each respondent in increasing item-id order, producing a fixed linear test form.

Usage

```
select_sequential(pers, item, R, admin, adj_mat = NULL)
```

Arguments

pers	A data frame of current respondent ability estimates.
item	A data frame of current item parameter estimates.
R	A respondent-by-item matrix of potential responses.
admin	An integer administration matrix; 0 indicates an item has not been administered to a respondent. See meow() for details.
adj_mat	An item-item adjacency matrix. See construct_adj_mat().

Value

An updated administration matrix with each respondent's next item marked as administered.

Examples

```
sim <- meow(select_sequential, update_theta_mle, data_simple_1pl,
            data_args = list(N_persons = 10, N_items = 10), fix = "item")
nrow(sim$results)
```

---

update_maths_garden *Elo-style updates of person and item parameters (Maths Garden).*

Description

Updates both person and item parameters following Klinkenberg, Straatemeier, and van der Maas (2011), "Computer adaptive practice of Maths ability using a new item response model for on the fly ability and difficulty estimation." Learning rates are tunable through K_theta and K_b.

Usage

```
update_maths_garden(pers, item, R, admin, K_theta = 0.1, K_b = 0.1)
```

Arguments

pers	A data frame of current respondent parameter estimates.
item	A data frame of current item parameter estimates.
R	A respondent-by-item matrix of potential responses.
admin	An integer administration matrix; non-zero entries indicate administered items. See meow() for details.
K_theta	Learning rate for person ability updates. Defaults to 0.1.
K_b	Learning rate for item difficulty updates. Defaults to 0.1.

Value

A list with two entries: pers and item, the data frames of updated respondent and item parameter estimates.

Examples

```
data <- data_simple_1pl(N_persons = 10, N_items = 10)
admin <- matrix(0L, 10, 10)
admin[, 1:5] <- 1L
R <- matrix(data$resp$resp, nrow = 10, byrow = TRUE)
upd <- update_maths_garden(data$pers_tru, data$item_tru, R, admin)
```

---

update_prowise_learn *Elo-style updates with paired item comparisons (Prowise Learn).*

Description

Updates both person and item parameters following Vermeiren et al. (2025), "Psychometrics of an Elo-based large-scale online learning system." Item difficulties are updated using paired comparisons of consecutively administered items, which controls the rating drift that can occur with naive Elo updates.

Usage

```
update_prowise_learn(pers, item, R, admin, K_theta = 0.1, K_b = 0.1)
```

Arguments

`pers`	A data frame of current respondent parameter estimates.
`item`	A data frame of current item parameter estimates.
`R`	A respondent-by-item matrix of potential responses.
`admin`	An integer administration matrix; non-zero entries indicate administered items. See `meow()` for details.
`K_theta`	Learning rate for person ability updates. Defaults to 0.1.
`K_b`	Learning rate for item difficulty updates. Defaults to 0.1.

Value

A list with two entries: `pers` and `item`, the data frames of updated respondent and item parameter estimates.

Examples

```
data <- data_simple_1pl(N_persons = 10, N_items = 10)
admin <- matrix(0L, 10, 10)
admin[, 1:5] <- 1L
R <- matrix(data$resp$resp, nrow = 10, byrow = TRUE)
upd <- update_prowise_learn(data$pers_tru, data$item_tru, R, admin)
```

update_theta_mle *Update person ability via maximum likelihood estimation.*

Description

This update function treats item parameters as fixed and known and updates person ability estimates after each iteration with a maximum likelihood estimate based on a 2PL item response function.

Usage

```
update_theta_mle(pers, item, R, admin)
```

Arguments

pers	A data frame of current respondent parameter estimates.
item	A data frame of item parameter values.
R	A respondent-by-item matrix of potential responses.
admin	An integer administration matrix; non-zero entries indicate administered items. See meow() for details.

Value

A list with two entries: pers, a data frame with updated respondent ability estimates, and item, the unchanged data frame of item parameters.

Examples

```
data <- data_simple_1pl(N_persons = 10, N_items = 10)
admin <- matrix(0L, 10, 10)
admin[, 1:5] <- 1L
R <- matrix(data$resp$resp, nrow = 10, byrow = TRUE)
upd <- update_theta_mle(data$pers_tru, data$item_tru, R, admin)
head(upd$pers)
```

Index