

Package: mirtverse (via r-universe)

October 29, 2026

Title A Friendly Workflow and Reporting Layer for 'mirt'

Version 0.1.0

Description Extends the 'mirt' package for item response theory with tidy outputs, 'ggplot2' graphics (trace lines, information, Wright maps, fit and local dependence plots), an automatic model selection assistant, plain-language interpretation of results, item health cards, one-call HTML reports, a point-and-click 'shiny' workbench that exports reproducible code, and a joint hierarchical model for responses and response times (van der Linden, 2007) with rapid-guessing flags.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0), mirt (>= 1.47)

Imports generics, ggplot2 (>= 3.4.0), patchwork, tibble, htmltools, shiny, bslib, DT, stats, utils, methods, grDevices, tools, base64enc, ragg

Suggests testthat (>= 3.0.0), ellmer (>= 0.3.0), jsonlite, keyring, knitr, rmarkdown, readxl, rstudioapi, withr, haven

Config/testthat/edition 3

VignetteBuilder knitr

URL <https://www.okanbulut.com/mirtverse/>,
<https://github.com/okanbulut/mirtverse>

BugReports <https://github.com/okanbulut/mirtverse/issues>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libx11-dev zlib1g-dev

Repository <https://okanbulut.r-universe.dev>

Date/Publication 2026-10-29 21:53:59 UTC

RemoteUrl <https://github.com/okanbulut/mirtverse>

RemoteRef HEAD
RemoteSha 5df5bde332adb16a46b700bca17b964ed4ceafab

Contents

augment_mirt	2
autoplot.SingleGroupClass	3
dif	4
fit_rt	5
gg_dif	7
gg_empirical	8
gg_info	9
gg_itemfit	9
gg_q3	10
gg_scores	11
gg_trace	11
gg_wright	12
glance_mirt	13
interpret.mv_dif	13
interpret_prompt	15
item_card	15
item_cards	16
item_table	17
launch_workbench	17
mv_ask	18
mv_ask_setup	20
mv_plan	21
mv_report	22
profile_data	23
run_script	24
select_model	25
sim_rt	26
theme_mirtverse	27
tidy_mirt	27
Index	29

augment_mirt	<i>Add person scores to the response data</i>
--------------	---

Description

Add person scores to the response data

Usage

```
## S3 method for class 'SingleGroupClass'
augment(x, data = NULL, method = "EAP", person_fit = FALSE, ...)
```

Arguments

x	a model returned by <code>mirt::mirt()</code> .
data	optional data frame to bind the scores to. Defaults to the response data used to fit the model.
method	scoring method passed to <code>mirt::fscores()</code> (e.g. "EAP", "MAP", "ML", "WLE").
person_fit	logical; add the standardized log-likelihood person-fit statistic Zh.
...	further arguments passed to <code>mirt::fscores()</code> .

Value

A tibble with the data plus .F1 (one column per factor), .se_F1, and optionally .Zh.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
head(augment(mod))
```

autoplot.SingleGroupClass

Plot a mirt model with ggplot2

Description

A single entry point to the mirtverse graphics.

Usage

```
## S3 method for class 'SingleGroupClass'
autoplot(
  object,
  type = c("trace", "expected", "info", "iteminfo", "se", "wright", "itemfit", "q3",
    "scores"),
  ...
)
```

Arguments

object	a model returned by <code>mirt::mirt()</code> .
type	one of "trace", "expected", "info", "iteminfo", "se", "wright", "itemfit", "q3", or "scores".
...	passed to the matching <code>gg_*()</code> function.

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
ggplot2::autoplot(mod, "wright")
```

dif

Differential item functioning (DIF) in one call

Description

Tests every item for DIF between groups (for example by gender or language) and reports the results as a tidy table with effect sizes. The analysis uses the constrained-baseline approach: a multiple-group model is fitted with all item parameters equal across groups and free latent means and variances, then each item's parameters are freed in turn and compared with a likelihood-ratio test (`mirt::DIF()` with `scheme = "drop"`).

Usage

```
dif(
  x,
  group = NULL,
  model = "2PL",
  items = NULL,
  which.par = NULL,
  p_adjust = "BH",
  alpha = 0.05,
  verbose = FALSE,
  ...
)
```

Arguments

<code>x</code>	a data frame or matrix of item responses, or a <code>mirt::multipleGroup()</code> model whose item parameters are constrained to be equal across groups (the baseline model).
<code>group</code>	the grouping variable: a vector with one value per respondent, or the name of a column of <code>x</code> (that column is then not treated as an item). Ignored when <code>x</code> is a model.
<code>model</code>	item type passed to <code>mirt::multipleGroup()</code> , e.g. "Rasch", "2PL", "3PL", "graded" or "gpcm". Ignored when <code>x</code> is a model.
<code>items</code>	items to test, by name or position. Defaults to all items; untested items act as anchors.

<code>which.par</code>	parameters to test. Defaults to all estimated item parameters (for example <code>a1</code> and <code>d</code> for the 2PL).
<code>p_adjust</code>	p-value adjustment across items (see <code>stats::p.adjust()</code>).
<code>alpha</code>	significance level for flagging an item, applied to the adjusted p-values.
<code>verbose</code>	logical; show progress.
<code>...</code>	further arguments passed to <code>mirt::multipleGroup()</code> .

Details

Effect sizes are reported for two-group comparisons as the difference in expected item score between the focal and the reference group at the same ability, averaged over the focal group's ability distribution: `esd` is the signed difference (positive values mean the item is easier for, or favours, the focal group) and `uesd` the unsigned difference, which also captures crossing (non-uniform) DIF. Both are on the item score scale, so for 0/1 items they are differences in the probability of a correct answer.

Value

An object of class `mv_dif` with elements `table` (one row per tested item: `item`, `X2`, `df`, `p.value`, `p.adjusted`, `flag`, `esd`, `uesd`, `converged`), `baseline` (the constrained model), `models` (the models with each item freed), `groups`, `reference`, `focal`, `which.par`, `alpha` and `p_adjust`. Use `tidy()`, `gg_dif()` and `interpret()` on it.

Examples

```
set.seed(1)
a <- matrix(rlnorm(8, 0.2, 0.3)); d <- matrix(rnorm(8))
d2 <- d; d2[1] <- d2[1] - 1
data <- rbind(simdata(a, d, 400, itemtype = "2PL"),
             simdata(a, d2, 400, itemtype = "2PL"))
gender <- rep(c("F", "M"), each = 400)
res <- dif(data, group = gender)
res
gg_dif(res)
interpret(res)
```

fit_rt

Joint model for responses and response times

Description

Fits van der Linden's (2007) hierarchical framework on top of a fitted **mirt** model. Response times follow a lognormal model,

$$\log T_{ij} \sim N(\beta_j - \tau_i, \alpha_j^{-2}),$$

where β_j is the time intensity of item j , α_j its time discrimination and τ_i the speed of person i . Ability and speed are bivariate normal with correlation ρ .

Usage

```
fit_rt(
  x,
  rt,
  flag_method = c("residual", "nt10"),
  z_crit = 1.96,
  quadpts = c(41, 21),
  tol = 1e-04,
  maxit = 500,
  verbose = FALSE
)
```

Arguments

<code>x</code>	a unidimensional model from <code>mirt::mirt()</code> fitted to the item responses.
<code>rt</code>	a matrix or data frame of response times (in seconds, or any positive unit) with the same rows and columns as the response data. Missing values are allowed.
<code>flag_method</code>	"residual" or "nt10"; see Details.
<code>z_crit</code>	critical value for the residual method.
<code>quadpts</code>	number of quadrature points per dimension.
<code>tol</code>	convergence tolerance (largest parameter change).
<code>maxit</code>	maximum number of EM iterations.
<code>verbose</code>	logical; print the log-likelihood during estimation.

Details

The item response parameters are taken from `x` and held fixed; the response-time parameters, the speed variance and the ability-speed correlation are estimated by marginal maximum likelihood with an EM algorithm on a two-dimensional quadrature grid. Persons are then scored jointly, so response times act as collateral information for ability. Standard errors come from the cross-product of the individual score contributions.

Rapid responses are flagged in two ways: "residual" marks responses that are much faster than expected for that person on that item (standardized log-time residual below `-z_crit`); "nt10" uses the normative threshold of 10% of the item's mean time (Wise & Ma, 2012). Response time effort (RTE; Wise & Kong, 2005) is the proportion of a person's responses that are not flagged. The l_t person-fit statistic (van der Linden & Guo, 2008) flags aberrant overall timing.

Value

An object of class `mv_rt` with elements `items` (time intensity and discrimination with SEs), `population` (speed SD and ability-speed correlation), `persons` (joint ability and speed estimates, RTE and l_t), `flags` (logical matrix of rapid responses), `logLik`, `iterations` and `converged`. Use `tidy()`, `glance()`, `augment()`, `autoplot()` and `interpret()` on it.

References

- van der Linden, W. J. (2007). A hierarchical framework for modeling speed and accuracy on test items. *Psychometrika*, 72, 287-308.
- van der Linden, W. J., & Guo, F. (2008). Bayesian procedures for identifying aberrant response-time patterns in adaptive testing. *Psychometrika*, 73, 365-384.
- Wise, S. L., & Kong, X. (2005). Response time effort: A new measure of examinee motivation in computer-based tests. *Applied Measurement in Education*, 18, 163-183.
- Wise, S. L., & Ma, L. (2012). Setting response time thresholds for a CAT item pool: The normative threshold method. Paper presented at NCME.

Examples

```
sim <- sim_rt(N = 1000, J = 15, rho = -0.4, seed = 1)
mod <- mirt(sim$responses, 1, verbose = FALSE)
fit <- fit_rt(mod, sim$rt)
fit
tidy(fit)
```

gg_dif

DIF effect sizes plot

Description

type = "effect" shows the signed expected-score difference for each item, with flagged items highlighted; type = "curves" overlays the expected item score curves of each group for the chosen (by default the flagged) items.

Usage

```
gg_dif(
  x,
  type = c("effect", "curves"),
  which.items = NULL,
  theta_lim = c(-4, 4)
)
```

Arguments

- | | |
|-------------|--|
| x | an mv_dif object from <code>dif()</code> . |
| type | "effect" or "curves". |
| which.items | items to draw for type = "curves", by name. Defaults to the flagged items, or the item with the largest effect if none is flagged. |
| theta_lim | range of the latent trait for type = "curves". |

Value

A ggplot object.

Examples

```
set.seed(1)
a <- matrix(rlnorm(8, 0.2, 0.3)); d <- matrix(rnorm(8))
d2 <- d; d2[1] <- d2[1] - 1
data <- rbind(simdata(a, d, 400, itemtype = "2PL"),
              simdata(a, d2, 400, itemtype = "2PL"))
res <- dif(data, group = rep(c("F", "M"), each = 400))
gg_dif(res)
gg_dif(res, type = "curves")
```

gg_empirical

Empirical versus model-implied item curve

Description

Groups examinees by their ability estimate and compares the observed mean item score in each group (with 95% intervals) with the model-implied curve. Large departures point to misfit.

Usage

```
gg_empirical(x, item, bins = 10, theta_lim = c(-3, 3))
```

Arguments

x	a unidimensional model returned by <code>mirt::mirt()</code> .
item	a single item, by position or name.
bins	number of ability groups.
theta_lim	range of the latent trait.

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
gg_empirical(mod, 1)
```

gg_info	<i>Test and item information curves</i>
---------	---

Description

Test and item information curves

Usage

```
gg_info(  
  x,  
  type = c("test", "items", "se"),  
  which.items = NULL,  
  theta_lim = c(-4, 4)  
)
```

Arguments

x	a unidimensional model returned by <code>mirt::mirt()</code> .
type	"test" for the test information function with its conditional standard error, "items" for item information curves, or "se" for the conditional standard error of measurement alone.
which.items	items to include for type = "items".
theta_lim	range of the latent trait.

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)  
gg_info(mod)  
gg_info(mod, type = "items")
```

gg_itemfit	<i>Item fit plot</i>
------------	----------------------

Description

Shows the RMSEA of the S-X2 item-fit statistic for each item and marks items whose p-value falls below alpha.

Usage

```
gg_itemfit(x, alpha = 0.01, p_adjust = "none")
```

Arguments

`x` a model returned by `mirt::mirt()`.
`alpha` significance level for flagging misfit.
`p_adjust` p-value adjustment (see `stats::p.adjust()`).

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
gg_itemfit(mod)
```

gg_q3

Local dependence heatmap (Yen's Q3)

Description

Local dependence heatmap (Yen's Q3)

Usage

```
gg_q3(x, cutoff = 0.2, label = TRUE)
```

Arguments

`x` a model returned by `mirt::mirt()`.
`cutoff` absolute residual correlation above the average Q3 that is marked as possible local dependence (0.2 by default, Christensen et al., 2017).
`label` logical; print values in flagged cells.

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
gg_q3(mod)
```

`gg_scores`*Distribution of person scores*

Description

Distribution of person scores

Usage

```
gg_scores(x, method = "EAP")
```

Arguments

<code>x</code>	a model returned by <code>mirt::mirt()</code> .
<code>method</code>	scoring method passed to <code>mirt::fscores()</code> .

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
gg_scores(mod)
```

`gg_trace`*Item characteristic (trace) curves*

Description

Item characteristic (trace) curves

Usage

```
gg_trace(  
  x,  
  which.items = NULL,  
  type = c("prob", "expected"),  
  facet = TRUE,  
  theta_lim = c(-4, 4)  
)
```

Arguments

x	a unidimensional model returned by <code>mirt::mirt()</code> .
which.items	items to plot, by position or name. Defaults to all.
type	"prob" for category probability curves or "expected" for the expected item score.
facet	logical; one panel per item. If FALSE all items share one panel (only sensible for type = "expected" or dichotomous items).
theta_lim	range of the latent trait.

Value

A ggplot object.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
gg_trace(mod)
gg_trace(mod, facet = FALSE)
```

gg_wright	<i>Wright map (person-item map)</i>
-----------	-------------------------------------

Description

Places the distribution of person estimates beside the item locations on the same latent scale, so you can see whether the test is targeted at the examinees. Polytomous items show each threshold.

Usage

```
gg_wright(x, method = "EAP", thresholds = TRUE, bins = 30)
```

Arguments

x	a unidimensional model returned by <code>mirt::mirt()</code> .
method	scoring method for persons, passed to <code>mirt::fscores()</code> .
thresholds	logical; plot every threshold of polytomous items rather than their average location.
bins	number of histogram bins for persons.

Value

A patchwork object (a combined ggplot).

Examples

```
mod <- mirt(Science, 1, verbose = FALSE)
gg_wright(mod)
```

glance_mirt	<i>One-row model summary</i>
-------------	------------------------------

Description

Collects sample size, information criteria, convergence, limited-information fit (M2 or C2) and marginal reliability in a single row, which makes it easy to compare models with `rbind()`.

Usage

```
## S3 method for class 'SingleGroupClass'
glance(x, fit = TRUE, ...)

## S3 method for class 'MultipleGroupClass'
glance(x, ...)
```

Arguments

<code>x</code>	a model returned by <code>mirt::mirt()</code> .
<code>fit</code>	logical; compute the M2-family fit statistics. For polytomous items the C2 statistic is used.
<code>...</code>	unused.

Value

A one-row tibble.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
glance(mod)
```

interpret.mv_dif	<i>Plain-language interpretation of IRT results</i>
------------------	---

Description

Turns a fitted model into short, readable statements about model fit, reliability, test targeting, problem items, local dependence and person fit. Each statement quotes the numbers it rests on, so the text can go straight into a report. The rules used are documented in the Details.

Usage

```
## S3 method for class 'mv_dif'
interpret(x, ...)

interpret(x, ...)

## S3 method for class 'SingleGroupClass'
interpret(x, alpha = 0.01, ...)

## S3 method for class 'mv_model_selection'
interpret(x, ...)

## S3 method for class 'mv_rt'
interpret(x, ...)
```

Arguments

x	a model from <code>mirt::mirt()</code> , an <code>mv_model_selection</code> object from <code>select_model()</code> , or an <code>mv_rt</code> object from <code>fit_rt()</code> .
...	unused.
alpha	significance level for item misfit.

Details

Rules of thumb used:

- RMSEA (M2/C2): $\leq .05$ close, $\leq .08$ reasonable, $\leq .10$ mediocre, otherwise poor.
- Marginal reliability: $\geq .90$ excellent, $\geq .80$ good, $\geq .70$ acceptable.
- Discrimination: < 0.5 low, > 4 suspiciously high; location beyond ± 3 extreme; guessing $> .30$ high; $S-X^2$ $p < \alpha$ misfit.
- Local dependence: Q3 more than 0.2 above the average Q3.
- Person fit: $Z_h < -1.96$ flags aberrant response patterns.

Value

An object of class `mv_interpretation`: a named list of character vectors, one per section. It prints as formatted text; use `as.character()` or `format(x, "markdown")` to reuse it.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
interpret(mod)
```

interpret_prompt	<i>Build a prompt for an AI assistant from a fitted model</i>
------------------	---

Description

Packs the key results of a model (parameters, fit, reliability and the rule-based interpretation) into a text prompt you can paste into any large language model to get a longer narrative or a draft technical-report section. Nothing is sent anywhere by this function.

Usage

```
interpret_prompt(x, audience = "a psychometric technical report")
```

Arguments

x	a model from <code>mirt::mirt()</code> .
audience	who the text is for, e.g. "teachers" or "a psychometric technical report".

Value

A single character string.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
cat(interpret_prompt(mod))
```

item_card	<i>Item health card</i>
-----------	-------------------------

Description

A one-page visual summary of a single item for item review meetings: trace curves, information, observed versus expected scores, the response distribution and a key-statistics panel with a traffic-light status.

Usage

```
item_card(x, item, alpha = 0.01, tab = NULL)
```

Arguments

x	a unidimensional model from <code>mirt::mirt()</code> .
item	a single item, by position or name.
alpha	significance level for item misfit.
tab	optional precomputed <code>item_table()</code> (saves time when making many cards).

Details

Status rules: **red** for misfit, low discrimination ($a < 0.5$) or an item-total correlation below .10; **amber** for any other flag (see `item_table()`); **green** otherwise.

Value

A patchwork object.

Examples

```
mod <- mirt(Science, 1, verbose = FALSE)
item_card(mod, "Work")
```

item_cards

Health cards for many items

Description

Health cards for many items

Usage

```
item_cards(x, which.items = NULL, file = NULL, alpha = 0.01)
```

Arguments

x	a unidimensional model from <code>mirt::mirt()</code> .
which.items	items to include (default: all).
file	optional path to an .html file. When given, all cards are written to one self-contained page with a status overview table.
alpha	significance level for item misfit.

Value

A named list of cards (invisibly when file is given).

Examples

```
mod <- mirt(Science, 1, verbose = FALSE)
cards <- item_cards(mod)
item_cards(mod, file = tempfile(fileext = ".html"))
```

item_table	<i>Tidy item statistics and item fit in one table</i>
------------	---

Description

Combines classical item statistics (`mirt::itemstats()`), IRT parameters and item fit (`mirt::itemfit()`) into one row per item, with a flag column that marks items worth reviewing.

Usage

```
item_table(x, fit_stats = "S_X2", p_adjust = "none", alpha = 0.01)
```

Arguments

x	a model returned by <code>mirt::mirt()</code> .
fit_stats	fit statistic passed to <code>mirt::itemfit()</code> .
p_adjust	p-value adjustment method (see <code>stats::p.adjust()</code>).
alpha	significance level used for misfit flags.

Value

A tibble with one row per item.

Examples

```
mod <- mirt(expand.table(LSAT7), 1, verbose = FALSE)
item_table(mod)
```

launch_workbench	<i>Launch the mirtverse workbench</i>
------------------	---------------------------------------

Description

Opens a point-and-click application for a complete IRT analysis: load and score data, explore items, let mirtverse pick a model or choose one yourself, read the plain-language interpretation, inspect item health cards and graphics, score examinees, analyse response times, and download an HTML report. Every step is written into an R script you can download, so the analysis is reproducible and a good way to learn the R code.

Usage

```
launch_workbench(data = NULL, ...)
```



```
workbench_app(data = NULL)
```

Arguments

data optional data frame of item responses to start with.
 ... passed to `shiny::runApp()` (e.g. `launch.browser`, `port`).

Value

Runs the app; returns nothing useful.

Examples

```
if (interactive()) {
  launch_workbench()
  launch_workbench(expand.table(LSAT7))
}
```

mv_ask

*Write an analysis script from a plain-language request***Description**

Describe the analysis in your own words, for example "fit a 2PL, check DIF by gender and make a report", and `mv_ask()` writes a complete, commented R script for it. The request is turned into an analysis plan (see `mv_plan()`), checked against your data, and the script is built from tested templates, so it only uses `mirt` and `mirtverse` functions with checked arguments. Nothing is run until you call `run_script()`.

Usage

```
mv_ask(
  request = NULL,
  data = NULL,
  plan = NULL,
  provider = NULL,
  model = NULL,
  anonymize = FALSE,
  dry_run = FALSE,
  ask = interactive(),
  ...
)
```

Arguments

request a plain-language description of the analysis. To change an earlier script, pass it first: `mv_ask(scr, "use a 3PL instead")`.

data the data: a data frame, or the path of a `.csv`, `.xlsx`, `.sav`, `.dta`, `.sas7bdat` or `.rds` file. The script refers to a data frame by the name you pass, so pass a named object (e.g. `data = mydata`).

plan	an mv_plan() to use instead of a request.
provider	"claude", "openai", "ollama" or "none" (keyword parser). Defaults to the provider chosen with mv_ask_setup() , then to whichever API key is available, then to "none".
model	the AI model name; see mv_ask_setup() .
anonymize	logical; replace column names with V1, V2, ... (and drop category labels) before anything is sent to the AI model. The names are restored locally.
dry_run	logical; print exactly what would be sent to the AI model and return it, without sending anything or writing a script.
ask	logical; ask follow-up questions in the console when the keyword parser needs more information. If FALSE the questions are reported as an error instead.
...	unused; there to catch a mistyped argument such as plan.

Details

Requests are understood by an AI model set up with [mv_ask_setup\(\)](#) (Claude, OpenAI GPT or a local Ollama model), or, when none is set up, by a built-in keyword parser that works offline and asks follow-up questions in the console when something is unclear. The AI model only sees your request and a summary of the columns ([profile_data\(\)](#)); never individual responses. Use `dry_run = TRUE` to see exactly what would be sent.

Value

An object of class `mv_script` with elements `script` (the code as a character vector), `plan`, `profile` (see [profile_data\(\)](#)), `request`, `data_name` and `parser` (what understood the request). Printing it shows the plan, any assumptions and the code. With `dry_run = TRUE`, the text that would be sent, invisibly.

See Also

[run_script\(\)](#), [save_script\(\)](#), [mv_plan\(\)](#), [mv_ask_setup\(\)](#).

Examples

```
scr <- mv_ask("fit a graded response model, plot the curves and summarise",
             data = Science, provider = "none")

scr
scr <- mv_ask(scr, "also make a report")
mv_ask("check DIF by gender", data = Science, dry_run = TRUE)

res <- run_script(scr, echo = FALSE)
res$mod
```

mv_ask_setup

*Set up an AI model for mv_ask()***Description**

Chooses which AI model `mv_ask()` uses to understand plain-language requests and stores the API key so you only do this once. Three providers are supported through the `ellmer` package:

Usage

```
mv_ask_setup(
  provider = c("claude", "openai", "ollama", "none"),
  model = NULL,
  key = NULL,
  save = c("ask", "renviron", "keyring", "none"),
  test = TRUE
)
```

Arguments

<code>provider</code>	"claude", "openai", "ollama" or "none".
<code>model</code>	model name; defaults to "claude-opus-5-5" for Claude, the <code>ellmer</code> default for OpenAI and the first installed model for Ollama.
<code>key</code>	the API key. If NULL and none is set, you are asked for it with a masked prompt.
<code>save</code>	where to keep the key and settings for future sessions: "ask" (choose interactively), "renviron" (your <code>.Renviron</code> file), "keyring" (the system keychain, needs the <code>keyring</code> package) or "none" (this session only).
<code>test</code>	logical; send one tiny test request to check the setup.

Details

- "claude": Anthropic's Claude, with your own key from <https://console.anthropic.com> (stored as `ANTHROPIC_API_KEY`).
- "openai": OpenAI's GPT models, with your own key from <https://platform.openai.com> (stored as `OPENAI_API_KEY`).
- "ollama": a free model running on your own computer with Ollama (<https://ollama.com>). Install Ollama, run `ollama pull qwen2.5` (or another model) once, and nothing ever leaves your machine.

Use `provider = "none"` to always use the built-in keyword parser, which needs no AI model and no internet connection.

Only a summary of your columns (see `profile_data()`) and your request are sent; never individual responses. Keys are never written into scripts, reports or `mv_script` objects.

Value

The provider and model, invisibly.

Examples

```
## Not run:
mv_ask_setup("claude")
mv_ask_setup("ollama", model = "qwen2.5")

## End(Not run)
```

mv_plan

An analysis plan for mv_ask()

Description

A plan lists, in a structured form, what the generated script should do: which columns are items, which model to fit, which group to test for DIF and which outputs to produce. `mv_ask()` fills a plan from a plain-language request, but a plan can also be written by hand and passed to `mv_ask(plan = )`, or changed with `update_plan()`.

Usage

```
mv_plan(
  items = NULL,
  model = "auto",
  nfact = 1,
  group = NULL,
  missing = NULL,
  key = NULL,
  rt = NULL,
  steps = c("describe", "fitcheck", "interpret"),
  options = list(),
  assumptions = character()
)

update_plan(plan, ...)
```

Arguments

items	item columns: a character vector of names, a range such as "Q1:Q20", or NULL for all columns that look like items.
model	"auto" (compare models with <code>select_model()</code>), "Rasch", "2PL", "3PL", "graded" or "gpcm".
nfact	number of latent dimensions (1 to 3).
group	name of a grouping column (e.g. "gender") for a DIF analysis. Setting it adds the "dif" step.
missing	numeric codes that mean "missing", e.g. c(9, 99).
key	answer key for multiple-choice items that are not yet scored.

rt	response-time columns, one per item and in the same order. Setting it adds the "rt" step.
steps	the analysis steps, any of "describe", "fitcheck", "dif", "scores", "plots", "interpret", "cards", "report" and "rt". A model is always fitted.
options	per-step options as a named list of lists, e.g. <code>list(scores = list(method = "MAP", file = "scores.csv"), report = list(file = "report.html"), fit = list(criterion = "AIC"))</code> .
assumptions	notes about how the request was understood; they are printed with the plan and written as comments in the script.
plan	an <code>mv_plan</code> object.
...	fields to change, as in <code>mv_plan()</code> .

Value

An object of class `mv_plan`.

Examples

```
mv_plan(model = "2PL", group = "gender", steps = c("fitcheck", "report"))
```

mv_report

One-call HTML report

Description

Writes a self-contained HTML report (no internet connection or extra software needed to open it) covering the plain-language summary, model fit, item statistics and parameters, graphics, a sum-score conversion table, optional item health cards and the R code needed to reproduce the analysis.

Usage

```
mv_report(
  x,
  file = "mirtverse-report.html",
  title = "IRT analysis report",
  author = NULL,
  cards = NULL,
  alpha = 0.01,
  dif = NULL,
  code = NULL,
  open = interactive() && !isTRUE(getOption("mirtverse.dry_run"))
)
```

Arguments

x	a unidimensional model from <code>mirt::mirt()</code> , or an <code>mv_model_selection</code> object from <code>select_model()</code> (the report then adds the model comparison and describes the recommended model).
file	path of the .html file to write.
title	report title.
author	optional author line.
cards	logical; include an item health card for every item. Defaults to TRUE for tests of up to 40 items.
alpha	significance level for item misfit.
dif	optional <code>mv_dif</code> object from <code>dif()</code> ; adds a section on differential item functioning.
code	optional character vector of R code to show in the reproducibility section (the workbench fills this in automatically).
open	logical; open the report in a browser afterwards.

Value

The path to the report, invisibly.

Examples

```
mod <- mirt(Science, 1, verbose = FALSE)
f <- mv_report(mod, file = tempfile(fileext = ".html"))
```

profile_data

Summarise a data set without revealing its values

Description

Describes each column of a data set (type, number of distinct values, range, share missing) and guesses its role: an item (whole numbers with few categories), a group variable (few categories, e.g. gender), an id, a response time (rt), or other. This summary is the only information `mv_ask()` ever shares with an AI model; individual responses are never included. Category labels are kept only for likely grouping variables.

Usage

```
profile_data(data)
```

Arguments

data	a data frame.
------	---------------

Value

A tibble of class `mv_profile` with one row per column: `column`, `type`, `n_distinct`, `min`, `max`, `prop_missing`, `role` and `levels`. The number of rows of data is stored in the `n` attribute.

Examples

```
profile_data(Science)
```

```
run_script
```

```
Run or save a script written by mv_ask()
```

Description

`run_script()` runs the script in a fresh environment and returns the objects it created (for example `mod`, `sel`, `dif_res`, `scores`). With `check = TRUE` it first runs the whole script on simulated data with the same structure as yours (same items, categories and groups) in a temporary folder, so errors show up before your data are touched. `save_script()` writes the script to a `.R` file.

Usage

```
run_script(x, check = FALSE, echo = interactive())
```

```
save_script(x, file = "analysis.R", open = FALSE)
```

Arguments

<code>x</code>	an <code>mv_script</code> from <code>mv_ask()</code> .
<code>check</code>	logical; do a dry run on simulated data first.
<code>echo</code>	logical; print each line and its output as it runs.
<code>file</code>	path of the <code>.R</code> file to write.
<code>open</code>	logical; open the file in the RStudio or Positron editor (needs the <code>rstudioapi</code> package).

Value

`run_script()`: a named list of the objects the script created, invisibly. `save_script()`: the file path, invisibly.

Examples

```
scr <- mv_ask(data = Science, plan = mv_plan(model = "graded", steps = "fitcheck"))
f <- save_script(scr, tempfile(fileext = ".R"))

res <- run_script(scr, echo = FALSE)
```

select_model	<i>Model selection assistant</i>
--------------	----------------------------------

Description

Fits a set of candidate IRT models to the same data, compares them on information criteria, likelihood-ratio tests and limited-information fit, and recommends one with a plain-language explanation.

Usage

```
select_model(  
  data,  
  candidates = NULL,  
  dims = 1,  
  criterion = c("BIC", "AIC", "SABIC"),  
  fit = TRUE,  
  verbose = TRUE,  
  ...  
)
```

Arguments

data	a data frame or matrix of item responses.
candidates	optional character vector of itemtype values passed to <code>mirt::mirt()</code> .
dims	number(s) of latent dimensions to try.
criterion	information criterion used to rank models.
fit	logical; compute M2/C2 fit statistics for each model.
verbose	logical; print progress.
...	further arguments passed to <code>mirt::mirt()</code> .

Details

By default the candidates depend on the data: for dichotomous items the Rasch, 2PL and 3PL models; for polytomous (or mixed) items the partial credit (Rasch), generalized partial credit and graded response models. Set `dims = 1:2` to also try two-factor exploratory versions.

Value

An object of class `mv_model_selection` with elements `table` (comparison table), `models` (fitted models), `best` (the recommended model), `lrt` (likelihood-ratio tests between nested models) and `explanation` (text).

Examples

```
sel <- select_model(expand.table(LSAT7))
sel
sel$best
```

sim_rt

Simulate responses and response times

Description

Generates data from a 2PL model for responses and the lognormal model for response times, with correlated ability and speed. Useful for teaching and for checking [fit_rt\(\)](#).

Usage

```
sim_rt(N = 1000, J = 20, rho = -0.3, sd_speed = 0.3, rapid = 0, seed = NULL)
```

Arguments

N	number of persons.
J	number of items.
rho	correlation between ability and speed.
sd_speed	standard deviation of speed.
rapid	proportion of persons who rapid-guess on a random 30% of the items (fast responses, correct at chance level .25).
seed	optional random seed.

Value

A list with responses, rt (seconds), theta, tau, rapid (logical matrix of simulated rapid guesses) and the true item parameters a, d, beta, alpha.

Examples

```
sim <- sim_rt(N = 200, J = 10, seed = 1)
str(sim, max.level = 1)
```

theme_mirtverse	<i>A clean ggplot2 theme for psychometric graphics</i>
-----------------	--

Description

A clean ggplot2 theme for psychometric graphics

Usage

```
theme_mirtverse(base_size = 11)
```

Arguments

base_size base font size.

Value

A ggplot2 theme.

tidy_mirt	<i>Tidy a mirt model into a data frame of parameters</i>
-----------	--

Description

Returns one row per estimated parameter, ready for dplyr, tables, and plotting. For unidimensional models the traditional IRT parameterization (discrimination a, difficulty b, guessing g, upper bound u) is used by default; multidimensional models are reported in slope-intercept form.

Usage

```
## S3 method for class 'SingleGroupClass'
tidy(
  x,
  IRTpars = TRUE,
  conf.int = FALSE,
  conf.level = 0.95,
  group_pars = FALSE,
  ...
)

## S3 method for class 'MultipleGroupClass'
tidy(
  x,
  IRTpars = TRUE,
  conf.int = FALSE,
  conf.level = 0.95,
```

```
    group_pars = TRUE,  
    ...  
  )
```

Arguments

<code>x</code>	a model returned by <code>mirt::mirt()</code> or <code>mirt::multipleGroup()</code> .
<code>IRTpars</code>	logical; use the traditional IRT parameterization when possible. Ignored (treated as FALSE) for multidimensional models.
<code>conf.int</code>	logical; add Wald confidence limits (needs standard errors, i.e. the model fitted with <code>SE = TRUE</code>).
<code>conf.level</code>	confidence level for <code>conf.int</code> .
<code>group_pars</code>	logical; also return latent mean and (co)variance parameters.
<code>...</code>	unused.

Value

A tibble with columns `item`, `itemtype`, `term`, `estimate`, and `std.error` (plus `group`, `conf.low`, `conf.high` when relevant).

Examples

```
mod <- mirt(expand.table(LSAT7), 1, SE = TRUE, verbose = FALSE)  
tidy(mod)  
tidy(mod, conf.int = TRUE)
```

Index

`augment()`, 6
`augment.SingleGroupClass`
 (`augment_mirt`), 2
`augment_mirt`, 2
`autoplot()`, 6
`autoplot.SingleGroupClass`, 3

`dif`, 4
`dif()`, 7, 23

`fit_rt`, 5
`fit_rt()`, 14, 26

`gg_dif`, 7
`gg_dif()`, 5
`gg_empirical`, 8
`gg_info`, 9
`gg_itemfit`, 9
`gg_q3`, 10
`gg_scores`, 11
`gg_trace`, 11
`gg_wright`, 12
`glance()`, 6
`glance.MultipleGroupClass`
 (`glance_mirt`), 13
`glance.SingleGroupClass` (`glance_mirt`),
 13
`glance_mirt`, 13

`interpret` (`interpret.mv_dif`), 13
`interpret()`, 5, 6
`interpret.mv_dif`, 13
`interpret_prompt`, 15
`item_card`, 15
`item_cards`, 16
`item_table`, 17
`item_table()`, 15, 16

`launch_workbench`, 17

`mirt::DIF()`, 4

`mirt::fscores()`, 3, 11, 12
`mirt::itemfit()`, 17
`mirt::itemstats()`, 17
`mirt::mirt()`, 3, 6, 8–17, 23, 25, 28
`mirt::multipleGroup()`, 4, 5, 28
`mv_ask`, 18
`mv_ask()`, 20, 21, 23, 24
`mv_ask_setup`, 20
`mv_ask_setup()`, 19
`mv_plan`, 21
`mv_plan()`, 18, 19
`mv_report`, 22

`profile_data`, 23
`profile_data()`, 19, 20

`run_script`, 24
`run_script()`, 18, 19

`save_script` (`run_script`), 24
`save_script()`, 19
`select_model`, 25
`select_model()`, 14, 21, 23
`shiny::runApp()`, 18
`sim_rt`, 26
`stats::p.adjust()`, 5, 10, 17

`theme_mirtverse`, 27
`tidy()`, 5, 6
`tidy.MultipleGroupClass` (`tidy_mirt`), 27
`tidy.SingleGroupClass` (`tidy_mirt`), 27
`tidy_mirt`, 27

`update_plan` (`mv_plan`), 21
`update_plan()`, 21

`workbench_app` (`launch_workbench`), 17