

Package: kits (via r-universe)

October 31, 2026

Title Psychometric Tools for the Kids in Taiwan Developmental Scale

Version 0.25.2

Date 2026-10-30

Description Tools for the developmental scale of KIT (Kids in Taiwan: National Longitudinal Study of Child Development and Care). Each model is written once and 'RTMB' gives every derivative (the conditional maximum likelihood, the person scores, the threshold standard errors, the norms); the closed-form steps use base R, and the data are handled with 'dplyr', 'tidyr' and 'purrr'. Items are calibrated with the partial credit model (Masters, 1982) <[doi:10.1007/BF02296272](https://doi.org/10.1007/BF02296272)> by conditional maximum likelihood, written as the elementary symmetric functions of the step parameters. Cohorts are linked by mean/sigma (Marco, 1977) <[doi:10.1111/j.1745-3984.1977.tb00033.x](https://doi.org/10.1111/j.1745-3984.1977.tb00033.x)> on the Thurstonian thresholds, and anchor items link to the D score of the Global Scales for Early Development (Weber et al., 2019) <[doi:10.1136/bmjgh-2019-001724](https://doi.org/10.1136/bmjgh-2019-001724)>. Person scores are weighted likelihood estimates (Warm, 1989) <[doi:10.1007/BF02294627](https://doi.org/10.1007/BF02294627)>; for the partial credit model they equal the Jeffreys modal estimates (Firth, 1993) <[doi:10.1093/biomet/80.1.27](https://doi.org/10.1093/biomet/80.1.27)> (Magis, 2015) <[doi:10.1007/s11336-013-9378-5](https://doi.org/10.1007/s11336-013-9378-5)>, which are computed by the Newton method of 'RTMB', one per item set and raw score. Norms are continuous sinh-arcsinh norms (Jones and Pewsey, 2009) <[doi:10.1093/biomet/asp053](https://doi.org/10.1093/biomet/asp053)> fitted by maximum likelihood with the polynomial model and bounds of 'cNORM' (Lenhard and Lenhard, 2021) <[doi:10.1177/0013164420928457](https://doi.org/10.1177/0013164420928457)>, minimised by nlminb with the gradient and Hessian of 'RTMB'. Differential item functioning combines the likelihood-ratio test of Andersen (1973) <[doi:10.1007/BF02291180](https://doi.org/10.1007/BF02291180)> with the item-pair test of Bechger and Maris (2015) <[doi:10.1007/s11336-014-9408-y](https://doi.org/10.1007/s11336-014-9408-y)>, computed by CML in each group. The linking error at each score and through a chain of links, the statistics of the waves as a vertical scale, the precision of each form across the scale, and IRT true-score and observed-score equating of raw scores follow Kolen and Brennan (2014)

<doi:10.1007/978-1-4939-0317-7>. The plots are drawn with 'plotly'. The algorithms are also exported as plain functions (the 'algo_' layer) that chain with the pipe, and kits_algorithm() names the steps, formulas and references behind a result; teaching scripts in 'inst/scripts' recompute the estimates with 'RTMB' on the demo data. The companion package 'kitsPreview' computes the same results with 'dexter' and 'cNORM' for validation.

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

Language en-US

Depends R (>= 4.1.0)

Imports RTMB, dplyr (>= 1.1.0), graphics, grDevices, htmlwidgets, plotly, purrr (>= 1.0.0), stats, tidyr (>= 1.3.0), tools, utils, vctrs (>= 0.6.0)

Suggests cNORM, dexter, kitsPreview, knitr, quarto, rmarkdown, testthat (>= 3.0.0), webshot2

VignetteBuilder quarto

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Collate 'ability.R' 'algo.R' 'algorithm.R' 'anchors.R' 'as_data_frame.R' 'calibrate.R' 'cml.R' 'conditions.R' 'data.R' 'diagnostics.R' 'dif.R' 'drift.R' 'esf.R' 'fit.R' 'info.R' 'item_map.R' 'item_table.R' 'items.R' 'kits-package.R' 'kits.R' 'labels.R' 'line.R' 'link.R' 'link_check.R' 'link_se.R' 'link_steps.R' 'methods.R' 'norms.R' 'norms_screen.R' 'norms_shash.R' 'norms_steps.R' 'pcm_core.R' 'style.R' 'plots.R' 'plots_results.R' 'precision.R' 'print.R' 'resp.R' 'restscore.R' 'scale.R' 'score.R' 'simulate.R' 'stats.R' 'utils.R' 'vctrs.R' 'vertical.R'

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev

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

Date/Publication 2026-10-29 20:27:55 UTC

RemoteUrl https://github.com/jiwenTsai/kits

RemoteRef HEAD

RemoteSha 6df7a02b86825155a034e830f93d8623d20b0d5f

Contents

AGE_TICKS	4
algo_cml	4
algo_esf	5
algo_jme	6
algo_line	7
algo_link_error	8
algo_lw	9
algo_mean_sigma	10
algo_pcm	11
algo_screen	12
algo_shash	13
algo_shash_z	14
algo_thresholds	15
as.data.frame.kits	15
kitm03	17
kits	19
kits_algorithm	20
kits_anchors	21
kits_check	22
kits_convert	24
kits_dif	25
kits_dif_combine	28
kits_drift	29
kits_esf	30
kits_info	32
kits_information	33
kits_item	35
kits_item_restscore	38
kits_item_steps	39
kits_item_table	41
kits_itemfit	44
kits_link_check	45
kits_link_steps	47
kits_linking	49
kits_methods	51
kits_norm_ages	53
kits_norm_screen	54
kits_norm_steps	56
kits_norm_z	58
kits_norms	59
kits_pcm_prob	61
kits_precision	63
kits_resp	65
kits_save_plot	66
kits_sim	67
kits_subj	68

kits_subj_table	70
kits_vertical	71
NORM_PCT	73
plot.kits_norms	73
print.kits_link	75
print.kits_norm_screen	76
print.kits_table	76
summary.kits	78

Index	80
--------------	-----------

AGE_TICKS	<i>Age ticks for plots</i>
-----------	----------------------------

Description

The default age_ticks of the plots against age: plot() of a kits_item_table(), of a kits_norms() by group and of a kits_vertical().

Usage

AGE_TICKS

Format

A numeric vector: 3, 6, 12, 18, 24, 36, 48, 60, 72, 84, 96.

Examples

AGE_TICKS

algo_cml	<i>Conditional maximum likelihood of the PCM</i>
----------	--

Description

Maximizes the conditional likelihood of the partial credit model from the sufficient statistics, with the gradient and Hessian from RTMB. It does not track scales or check its input; kits_cml() and kits_item() do.

Usage

algo_cml(stats)

Arguments

stats The sufficient statistics: a kits_stats() result (or item\$stats).

Value

A list: coef (one row per step: item_id, item_score, beta, se), betas (the betas as a named list, one vector per item), vcov (the covariance of the betas), nll (-log L) and optim (the convergence record).

See Also

Other algo: [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
cml <- kits_stats(X) |>
  algo_cml()
cml$coef
```

 algo_esf

Elementary symmetric functions and the conditional item-score distributions

Description

For one set of items, the elementary symmetric functions gamma_r of CML and, for each item, the distribution of its score given the raw score r, with its mean and variance. It does not track scales or check its input; [kits_esf\(\)](#) does.

Usage

```
algo_esf(betas)
```

Arguments

betas The step parameters: a named list, one numeric vector per item (in step order).

Value

A list: items, m (steps per item), max_score, centre, log_gamma (r = 0..R), log_gamma_rest (one row per item: gamma without that item), prob (one matrix per item: $P(X_i = x | r)$, rows r, columns x), expected, variance and free (rows r, columns items).

See Also

Other algo: [algo_cml\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
esf <- list(a1 = c(-1, 0.5), a2 = 0.3) |>
  algo_esf()
exp(esf$log_gamma)
esf$prob$a1
```

algo_jme

Jeffreys modal estimates of ability and their SEs

Description

Jeffreys modal estimate: $\theta\text{-hat} = \text{argmax } \log L(\theta) + 1/2 \log I(\theta)$. For the PCM it equals Warm's (1989) WLE exactly (Firth, 1993; Magis, 2015), so `kits_subj()` reports it as the WLE. One estimate per combination of items and raw score, by RTMB's Newton method; finite also for a zero or a perfect score. $SE = 1 / \sqrt{I(\theta\text{-hat})}$, the asymptotic SE of the WLE (Warm, 1989). With two maxima (few items with very different difficulties) the higher one is taken (the value of the objective decides; it is found on a grid of θ , step 0.05); with two equally high maxima, the smaller θ . It does not track scales or check its input; `kits_subj()` and `kits_subj_table()` do.

Usage

```
algo_jme(
  betas,
  raw = 0:sum(lengths(betas)),
  items = rep(list(names(betas)), length(raw))
)
```

Arguments

<code>betas</code>	The step parameters: a named list, one numeric vector per item.
<code>raw</code>	The raw scores, one per row (default: 0 to the top score of all items).
<code>items</code>	The items answered, a list with one character vector per row (default: all items for every row).

Value

A data.frame with `theta` and `se`, one row per raw score.

References

Firth, D. (1993). Bias reduction of maximum likelihood estimates. *Biometrika*, 80(1), 27-38. [doi:10.1093/biomet/80.1.27](https://doi.org/10.1093/biomet/80.1.27)

Magis, D. (2015). A note on weighted likelihood and Jeffreys modal estimation of proficiency levels in polytomous item response models. *Psychometrika*, 80(1), 200-204. [doi:10.1007/s11336-01393785](https://doi.org/10.1007/s11336-01393785)

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, 54(3), 427-450. [doi:10.1007/BF02294627](https://doi.org/10.1007/BF02294627)

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5)
betas |>
  algo_jme()
algo_jme(betas, raw = c(1, 1), items = list(c("a1", "a2"), "a2"))
```

 algo_line

Apply, invert or compose straight lines

Description

A line is a list with intercept and slope (new value = intercept + slope x old value); a link is one too. A list of lines is composed in the order applied; `invert` gives the inverse; with `x` the line is applied. It does not track scales; [kits_convert\(\)](#) does.

Usage

```
algo_line(line, x = NULL, invert = FALSE)
```

Arguments

line	A line, or a list of lines (applied in order).
x	Optional values to convert.
invert	Use the inverse line, $x = (y - \text{intercept}) / \text{slope}$.

Value

Without `x`, the line (intercept, slope); with `x`, the converted values.

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
l1 <- list(intercept = 1, slope = 2)
l2 <- list(intercept = -1, slope = 0.5)
algo_line(l1, x = c(0, 1))
algo_line(l1, invert = TRUE)
list(l1, l2) |>
  algo_line(x = c(0, 1))
```

algo_link_error	<i>The linking error by resampling whole items</i>
-----------------	--

Description

Resamples the items `n_boot` times and redoes the screening and the mean/sigma line on each draw; the SDs of the intercept and slope over the draws are the linking error. It does not track scales or check much; [kits_link_error\(\)](#) does.

Usage

```
algo_link_error(screen, n_boot = 200, seed = 1)
```

Arguments

<code>screen</code>	An algo_screen() result (its points and screening rule are used again).
<code>n_boot</code>	The number of item resamples.
<code>seed</code>	The random seed (NULL: not repeatable); the current random state is left as it is.

Value

A list: `intercept_error`, `slope_error`, `n_failed` (draws whose screening failed), `vcov` (2 x 2) and `draws` (one row per kept draw: `intercept`, `slope`).

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
from <- c(-2, -1.1, 0, 0.9, 2, 3.1, 0.5, 1.4)
to <- c(-1, 0, 1.1, 2, 3, 4.2, 1.5, 2.4)
err <- algo_screen(from, to, item = rep(1:4, 2)) |>
  algo_link_error(n_boot = 100)
err$intercept_error
```

algo_lw

*The raw-score distribution given theta (Lord-Wingersky)***Description**

$P(r \mid \theta)$ for a set of PCM items by the Lord-Wingersky recursion: adding item j , $f_j(x) = \sum_k f_{(j-1)}(x - k) P_{jk}$. It does not track scales or check its input; [kits_precision\(\)](#) does.

Usage

```
algo_lw(theta, betas)
```

Arguments

theta	Abilities (a numeric vector).
betas	The step parameters: a list, one numeric vector per item.

Value

A matrix, one row per theta, one column per raw score 0..R.

References

Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. L. (1995). Item response theory for scores on tests including polytomous items with ordered responses. *Applied Psychological Measurement*, 19(1), 39-49. doi:[10.1177/014662169501900105](https://doi.org/10.1177/014662169501900105)

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
P <- list(a1 = c(-1, 0), a2 = 0.5) |>
  algo_lw(theta = c(-1, 0, 1))
P
rowSums(P)
```

algo_mean_sigma	<i>The mean/sigma line</i>
-----------------	----------------------------

Description

The line new value = intercept + slope x old value with slope = SD(to) / SD(from) and the means aligned. It does not track scales or check its input; [kits_link_line\(\)](#) does.

Usage

```
algo_mean_sigma(from, to)
```

Arguments

from, to	The points on the two scales (numeric vectors of the same length). from can also be an algo_screen() result: then the kept points are used and to is not needed.
----------	--

Value

A list with intercept and slope.

References

Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd ed.). Springer.
[doi:10.1007/9781493903177](https://doi.org/10.1007/9781493903177)

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```
from <- c(-2, -1, 0, 1, 2, 3)
to <- c(-1, 0, 1, 2, 3.2, 6)
algo_mean_sigma(from, to)
algo_screen(from, to) |>
  algo_mean_sigma()
```

algo_pcm

*PCM category probabilities and moments of one item***Description**

The partial credit model for one item at abilities theta: the category probabilities, the expected score and the variance (the item information). It does not track scales or check its input; `kits_pcm_prob()` does.

Usage

```
algo_pcm(theta, beta, third = FALSE)
```

Arguments

theta	Abilities (a numeric vector; also an RTMB advector).
beta	The step parameters of the item, in step order.
third	Also return the third central moment M3.

Value

A list: P (a matrix, one row per theta, one column per score 0..m), E, Var, log_S (the log of the normalizing sum), and M3 when third = TRUE.

References

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149-174.
[doi:10.1007/BF02296272](https://doi.org/10.1007/BF02296272)

See Also

Other algo: `algo_cml()`, `algo_esf()`, `algo_jme()`, `algo_line()`, `algo_link_error()`, `algo_lw()`, `algo_mean_sigma()`, `algo_screen()`, `algo_shash()`, `algo_shash_z()`, `algo_thresholds()`

Examples

```
p <- c(-1, 0, 1) |>
  algo_pcm(beta = c(-0.5, 0.5))
p$P
p$E
```

 algo_screen

One round of screening of the link points

Description

Fits the mean/sigma line to all points and flags the points whose residual exceeds cut times the spread; with `unit = "item"` every point of a flagged item is dropped. It does not track scales or check much; [kits_link_screen\(\)](#) does.

Usage

```
algo_screen(
  from,
  to,
  item = seq_along(from),
  cut = 2,
  spread = "sd",
  unit = "step",
  se_from = NULL,
  se_to = NULL
)
```

Arguments

<code>from, to</code>	The points on the two scales (numeric vectors of the same length).
<code>item</code>	The item of each point (default: one item per point).
<code>cut</code>	The cutoff, in units of the spread.
<code>spread</code>	"sd" (the SD of the residuals), "mad" (1.4826 x their MAD) or "se" (the SE of each point from <code>se_from</code> and <code>se_to</code>).
<code>unit</code>	"step" (drop the flagged points) or "item" (drop every point of a flagged item).
<code>se_from, se_to</code>	The SE of each point (only for <code>spread = "se"</code> ; <code>se_to</code> defaults to 0).

Value

A list: `from`, `to`, `item`, `resid`, `z`, `flagged`, `dropped` (one value per point), `intercept0`, `slope0` (the line of all points), `cut` and `rule` (`spread`, `unit`, and the SEs). Give it to [algo_mean_sigma\(\)](#) and [algo_link_error\(\)](#).

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_shash\(\)](#), [algo_shash_z\(\)](#), [algo_thresholds\(\)](#)

Examples

```

from <- c(-2, -1, 0, 1, 2, 3, 0.5, 1.5)
to <- c(-1, 0, 1, 2, 3, 4, 1.5, 5.5)
s <- algo_screen(from, to, spread = "mad")
s$dropped
s |>
  algo_mean_sigma()

```

algo_shash

A shash norm model by maximum likelihood

Description

Fits a sinh-arcsinh (shash) distribution of y whose four parameters (μ , σ , ϵ , δ) are polynomials in x , by `nlminb` with the gradient and Hessian from `RTMB`, from one start of δ . It does not track scales or check its input; `kits_norm_fit()` does (several starts).

Usage

```
algo_shash(x, y, degrees = c(5, 4, 3, 2), delta0 = 1)
```

Arguments

<code>x</code>	Transformed age (for example <code>sqrt(age_m)</code>); it is standardized inside.
<code>y</code>	The scores.
<code>degrees</code>	The degrees of the four polynomials.
<code>delta0</code>	The start value of δ .

Value

A list: the coefficients `mu_est`, `sigma_est`, `epsilon_est`, `delta_est`, `degrees`, `age_mean`, `age_sd` (of x), value ($-\log L$), and the convergence record (convergence and message of `nlminb`, `converged`, `grad_max`, `hessian_pd`, `at_bound`).

References

Jones, M. C., & Pewsey, A. (2009). Sinh-arcsinh distributions. *Biometrika*, 96(4), 761-780.
[doi:10.1093/biomet/asp053](https://doi.org/10.1093/biomet/asp053)

See Also

Other algo: `algo_cml()`, `algo_esf()`, `algo_jme()`, `algo_line()`, `algo_link_error()`, `algo_lw()`, `algo_mean_sigma()`, `algo_pcm()`, `algo_screen()`, `algo_shash_z()`, `algo_thresholds()`

Examples

```
set.seed(1)
age <- runif(600, 3, 36)
D <- 20 + 15 * log(age) + rnorm(600, 0, 3)
fit <- algo_shash(sqrt(age), D, degrees = c(3, 2, 1, 1), delta0 = 0.5)
fit$value
```

algo_shash_z

Same-age z-scores under a shash norm model

Description

The parameters of an [algo_shash\(\)](#) model at x (clamped as in the fit), and the z-score $\sinh(\delta \cdot \text{asinh}((y - \mu) / \sigma) + \epsilon)$ of each y . It does not track scales or check its input; [kits_norm_z\(\)](#) does (from a norm table).

Usage

```
algo_shash_z(fit, x, y)
```

Arguments

<code>fit</code>	An algo_shash() result.
<code>x</code>	Transformed age, as in the fit.
<code>y</code>	The scores, one per x .

Value

The z-scores (a numeric vector).

See Also

Other algo: [algo_cml\(\)](#), [algo_esf\(\)](#), [algo_jme\(\)](#), [algo_line\(\)](#), [algo_link_error\(\)](#), [algo_lw\(\)](#), [algo_mean_sigma\(\)](#), [algo_pcm\(\)](#), [algo_screen\(\)](#), [algo_shash\(\)](#), [algo_thresholds\(\)](#)

Examples

```
set.seed(1)
age <- runif(600, 3, 36)
D <- 20 + 15 * log(age) + rnorm(600, 0, 3)
z <- algo_shash(sqrt(age), D, degrees = c(3, 2, 1, 1), delta0 = 0.5) |>
  algo_shash_z(sqrt(age), D)
c(mean(z), sd(z))
```

algo_thresholds	<i>Thurstonian thresholds of the steps and their SEs</i>
-----------------	--

Description

For each step k of each item, the ability where $P(X \geq k \mid \theta) = rp$ (by `uniroot()`), and its SE by the delta method with the derivative from RTMB. It does not track scales or check its input; `kits_thresholds()` and `kits_item()` do.

Usage

```
algo_thresholds(cml, rp = 0.5)
```

Arguments

<code>cml</code>	An <code>algo_cml()</code> result (its <code>coef</code> and <code>vcov</code>).
<code>rp</code>	The response probability (0.5: RP50).

Value

A data.frame, one row per step: `item_id`, `item_score`, `beta`, `beta_se`, `thr`, `thr_se`.

See Also

Other algo: `algo_cml()`, `algo_esf()`, `algo_jme()`, `algo_line()`, `algo_link_error()`, `algo_lw()`, `algo_mean_sigma()`, `algo_pcm()`, `algo_screen()`, `algo_shash()`, `algo_shash_z()`

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
kits_stats(X) |>
  algo_cml() |>
  algo_thresholds()
```

<code>as.data.frame.kits</code>	<i>One-row summaries of kits results, ready to stack</i>
---------------------------------	--

Description

`as.data.frame()` turns a result that is a list into a plain data.frame with one row (or one row per wave, per method, per item pattern), so that several results can be stacked with `dplyr::bind_rows()` and compared, for example the links of every domain. The numbers are those of the object (the same as `print()` and `summary()`). `checks` is the most serious status of the diagnostics ("warn", "note" or "ok"; see `kits_check()`).

Usage

```
## S3 method for class 'kits_link'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_drift'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_link_check'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_info'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_item'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_norm_screen'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_norm_fit'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'kits_stats'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A kits result (see above).
row.names, optional	Not used (as in <code>base::as.data.frame()</code>).
...	Not used.

Details

The columns:

- A link (`kits_linking()`, `kits_link_line()`), one row: from, to (the scale labels; several from scales are joined with ", "), kind ("calibration", "anchors", or NA for your own points), intercept, slope, intercept_error, slope_error, cov_intercept_slope (their covariance over the draws), intercept0, slope0 (the line before screening), n_points, n_dropped, n_items, n_failed (resamples that could not be computed), cut, spread, unit, step ("all" or the step, as text), n_boot, seed, key (the dscore key of the anchors), checks. Values that do not apply are NA.
- `kits_drift()`, one row: scale (of ref), intercept, slope, n_items, n_judged, n_drift (judged items with a dropped step), n_fix (items suggested for fixing), n_steps, n_steps_judged, n_steps_dropped, cut, min_n, checks.

- `kits_link_check()`, one row per method: from, to, method, intercept, slope, basis, max_abs_diff (the largest |diff| over theta), max_abs_z, max_abs_z_both (NA without bootstrap draws), checks.
- `kits_info()` (the provenance of any result), one row: call, version, settings (as text, "name = value; ..."), n_rows, data_md5 (of the responses, or of the values), persons_md5, input (the name of the data object, when it was a data.frame or a kits_subj).
- `kits_item()`, one row: scale, n_persons, n_items, n_steps, n_patterns, n_fixed (fixed steps), n_left_out (fixed items left out), neg_loglik, convergence, max_gradient, checks.
- `kits()`, one row: the columns of `kits_item()`, then theta_mean, theta_sd, n_zero, n_perfect, n_misfit (persons by flag).
- `kits_norm_screen()`, one row per (cohort x) wave: its waves table.
- `kits_norm_fit()`, one row: n, degrees (as text), neg_loglik, convergence, converged, grad_max, n_at_bound.
- `kits_stats()`, one row per item pattern: pattern_id, n_items, items (as text), n, max_raw.

Value

A data.frame.

Examples

```
betas <- list(a1 = -2, a2 = -1, a3 = c(-0.5, 0.5), a4 = 0, a5 = c(0.5, 1.5), a6 = 2, a7 = 3)
young <- kits_sim(rnorm(500, 0), betas[1:5], seed = 1)
old <- kits_sim(rnorm(500, 2), betas[3:7], prefix = "q", seed = 2)
item_young <- kits_item(young, scale = "young cohort")
item_old <- kits_item(old, scale = "old cohort")
link <- kits_linking(item_old, item_young, n_boot = 100)
as.data.frame(link)

# Several links in one table
link_step1 <- kits_linking(item_old, item_young, step = 1, n_boot = 100)
dplyr::bind_rows(as.data.frame(link), as.data.frame(link_step1))

# The calibrations and their provenance
dplyr::bind_rows(as.data.frame(item_young), as.data.frame(item_old))
as.data.frame(kits_info(item_young))
```

Description

Simulated versions of the two Kids in Taiwan (KIT) cohorts, language and motor only, for examples and tutorials. No row comes from a real child. The response columns are those of the real KIT response file, renamed as everywhere in kits: `visit_id` (KIT: `person_id`), `subj_id` (KIT: `pid`) and `cohort` (KIT: `sample`). On the real data, run `rename(visit_id = person_id, subj_id = pid, cohort = sample)` once and the same code runs.

Usage

```
kitm03
```

```
kitm36
```

Format

Each is a list with two elements.

responses is a tibble with one row per child x wave x item answered:

visit_id Visit (person-wave) id: one child at one wave, for example "M03_c55df9b_M12".

subj_id Child id: 7 random hex characters, like a short git hash, for example "c55df9b".

cohort Cohort: "M03" (3 to 72 months) or "M36" (36 to 96 months).

sex "boy" or "girl".

wave_m Wave, in months.

age_m Age at the visit, in months.

item_id Item id, as in KIT.

item_score Score, 0 to the maximum of the item.

domain "lan" (language) or "motor".

anchors is a data.frame with the columns of [kits_anchors\(\)](#): item_id, gsed_item, key, tau, note.

An object of class list of length 2.

Details

Each cohort is a list of two tables:

- responses: the responses, in long format (see below).
- anchors: the anchor key for the items of this cohort, from [kits_anchors\(\)](#) (key gsed2406): KIT items matched to GSED items, with their GSED difficulty tau. Use it to link to D with [kits_linking\(items, anchors\)](#). kitm03 has 27 anchors, kitm36 12.

The data were simulated from aggregates of the real KIT data (data-raw/kit_demo.R); the waves at 60 and 72 months of kitm03 go beyond the real M03 follow-up (48 months) and are not observed. Details: [tutorial_algorithms/01_calibration.html](#).

Examples

```
library(dplyr)
kitm03$responses |>
  distinct(visit_id, wave_m) |>
  count(wave_m)
kitm03$anchors |>
  count(key)
```

 kits

Calibrate and score in one step

Description

`kits_item()` and then `kits_subj()` on the same data. Use it when no link is needed in between (for example a single cohort). With linking, call the two functions separately. `plot()` draws a Wright map: the abilities and the step thresholds on the same theta axis.

Usage

```
kits(data, ..., scale = NULL, persons = NULL)

## S3 method for class 'kits'
print(x, digits = 2, ...)

## S3 method for class 'kits'
plot(x, ...)
```

Arguments

<code>data</code>	For type = "observed": the responses (as in <code>kits_item()</code>).
<code>...</code>	Not used.
<code>scale</code>	A label for the scale this calibration defines, for example "lan M03". Always give it; by default the domain and cohort of a <code>kits_resp()</code> , the name of data, or "scale 1". With <code>fixed_params</code> that record a scale, that scale is used (another scale stops).
<code>persons</code>	For type = "observed": optional data.frame with <code>visit_id</code> and the column named by group, as in <code>kits_dif()</code> .
<code>x</code>	A kits.
<code>digits</code>	Decimals to print (only the printout is rounded; the object keeps full precision).

Details

Details: `kits_algorithm("cml")`, `kits_algorithm("wle")`, [tutorial_algorithms/01_calibration.html](#) and [02_scoring.html](#).

Value

A kits: a list with `item` (the `kits_item()` result), `subj` (the `kits_subj()` result), `scale`, `diagnostics` (those of `item` and `subj`) and `info`.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
k <- kits(X)
k
head(k$subj)
p <- plot(k)
if (interactive()) p
```

kits_algorithm

The algorithm behind a result

Description

Says, in about 30 lines, which algorithm produced a result: its steps, one formula per step, the functions that run the steps (the kits_ steps and the algo_ building blocks, as |> pipelines), key references, and where to read more (the tutorial page in tutorial_algorithms/ and the teaching script in inst/scripts, with its parts).

Usage

```
kits_algorithm(x)
```

Arguments

x A result: `kits_item()` or `kits()`, the scores of `kits_subj()` or `kits_convert()`, `kits_linking()`, `kits_norms()`, `kits_norm_screen()`, `kits_dif()`, `kits_drift()`, `kits_precision()` or `kits_information()`. Or a topic: "cml", "wle", "link", "norms", "norm_screen", "dif", "drift", "information".

Value

Invisibly, a kits_algorithm object (a list: topic, title, facts, steps, components, references, tutorial, script, parts); it prints the algorithm. For a result, facts is one line on the result itself (items, steps, persons, scale).

References

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47, 149-174.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, 54, 427-450.

Kolen, M. J., & Brennan, R. L. (2014). Test equating, scaling, and linking (3rd ed.). Springer.

Examples

```
kits_algorithm("cml")

betas <- list(a1 = c(-1, 0), a2 = 0.5, a3 = -0.5)
item <- kits_sim(rnorm(200), betas, seed = 1) |>
  kits_item(scale = "demo")
kits_algorithm(item)

# The teaching script and its parts
algo <- kits_algorithm("link")
system.file("scripts", algo$script, package = "kits")
algo$parts
```

kits_anchors

The anchor items of the link to D (GSED), for one dscore key

Description

The anchor items of KIT for the link to the D score of the Global Scales for Early Development (GSED): each row is a KIT item, the GSED item it is matched to, and the difficulty tau of that GSED item in a key of the dscore package (van Buuren et al., 2025). Give the table to [kits_linking\(\)](#) as to; use one key per link (the link records it). Details: [vignette\("kits-flow", package = "kits"\)](#), section 3.

Usage

```
kits_anchors(key = c("gsed2406", "gsed2510"))
```

Arguments

key The dscore key: "gsed2406" (default, 29 anchors) or "gsed2510" (8 anchors, another D metric; gives a warning of class kits_warning_anchors).

Value

A data.frame with one row per anchor: item_id (the KIT item), gsed_item (the GSED item), key, tau (on the D scale), note (the matching decision) and domain ("lan", "motor" or "cog").

References

van Buuren, S., Eekhout, I., McCray, G., Lancaster, G. A., Waldman, M. R., McCoy, D. C., Gladstone, M., et al. (2025). Enhancing comparability in early child development assessment with the D-score. *International Journal of Behavioral Development*, 49(4), 348-364. doi:10.1177/01650254241294033

van Buuren, S., Eekhout, I., Huizing, A., & Seiden, J. (2025). *dscore: D-score for child development*. R package. <https://CRAN.R-project.org/package=dscore>

See Also[kits_linking\(\)](#)**Examples**

```

head(kits_anchors())

# A link to D from a simulated calibration whose items carry KIT item ids
anchors <- kits_anchors()
ids <- anchors$item_id[1:8]
betas <- as.list((anchors$tau[1:8] - 55) / 10)
names(betas) <- ids
X <- kits_sim(rnorm(600, 0, 1.5), betas, seed = 1)
item <- kits_item(X, scale = "sim M03")
kits_linking(item, anchors)

```

kits_check

Diagnostics of one or more results

Description

Every main result of kits checks itself when it is made and keeps a diagnostics table (`x$diagnostics` for lists, `attr(x, "diagnostics")` for tables), one row per check; its printout ends with "Checks: OK" or the number of warnings and notes. `kits_check()` collects the tables of several results; an object without diagnostics gets a row saying so (a stacked [kits_subj\(\)](#) gets its diagnostics recomputed).

Usage

```
kits_check(..., all = FALSE)
```

Arguments

<code>...</code>	Results of kits, named or not (the names label the rows).
<code>all</code>	FALSE (the default) keeps only notes and warnings; TRUE keeps every check.

Details

The checks:

- [kits_item\(\)](#): convergence, gradient, Hessian, extreme estimates, category counts, item pattern sizes (note below 400 person-waves) and item fit (infit or outfit outside 0.6 to 1.4).
- [kits_subj\(\)](#): shares of zero, perfect and misfit scores, and the reliability of each wave.
- [kits_subj_table\(\)](#): person-waves who answered items outside the standard form of their wave.
- [kits_itemfit\(\)](#): whether the item parameters were calibrated on these data.

- Links (`kits_linking()`, `kits_link_steps`): points kept and dropped, the correlation of the kept points, the linking error, slope vs 1, the common-item share and the domain balance.
- `kits_link_check()`: a check method that differs from the link by more than 2 errors (or tol).
- `kits_item_table()` with data: items separated by age.
- `kits_norms()`: convergence, gradient, coefficients at their limits, the largest delta, and the same-age z by age decile.
- `kits_norm_screen()`: waves left out. `kits_drift()`: drifted items.
- `kits_dif()`: the LR test, the largest step difference, the overall DIF test, the largest ldifl, items not compared, steps not observed in a group.
- `kits_vertical()`: scale shrinkage, falling means, and cohorts that differ at a shared wave.
- `kits_precision()`: the usable range of each form.

Details: vignette("kits-flow", package = "kits"), section 6.

Value

A table (a data.frame), warnings first: object, and the diagnostics columns check, value, status ("ok", "note" or "warn") and message.

Quality control for a new wave

A short checklist, after Kolen and Brennan (2014; Table 8.8, p. 336, and p. 333):

1. Drift check before fixing: `kits_drift()` against the original calibration (pp. 288-289).
2. Fix to the original calibration: `kits_item(X_new, fixed_params = drift$fix)` (pp. 293-294).
3. Double link: a second route to the target scale (`kits_link_check()`; pp. 295-296).
4. Item pattern n: about 400 person-waves per form (p. 304; the check of `kits_item()`).
5. Common-item share: at least 20% of the items, a content miniature (`kits_linking()`).
6. Consistency with the other cohort at the same age (p. 333; `kits_vertical()`).
7. md5 provenance: compare the fingerprints of `kits_info()` (p. 333).

References

Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd ed.). Springer.
doi:10.1007/9781493903177

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
item <- kits_item(X)
s <- kits_subj(X, item)
kits_check(item, s)
kits_check(calibration = item, all = TRUE)
summary(item)
```

kits_convert

Convert scores or thresholds with one or more links

Description

Applies the links in order, each a straight line: new value = intercept + slope x old value, and new SE = slope x old SE. Before each link the scale of x must be one the link starts from (`link$from_scale`), else it stops (class `kits_error_scale`); an unrecorded scale warns. When every link has bootstrap draws (`n_boot > 0`), a table also gets `se_link` (linking error only) and `se_total = sqrt(se^2 + se_link^2)`.

Usage

```
kits_convert(x, ..., from = NULL)
```

Arguments

x	Scores or thresholds: a <code>kits_subj()</code> result or a <code>data.frame</code> with <code>theta</code> (and <code>se</code>); a <code>kits_subj_table()</code> ; a <code>kits_item()</code> or its <code>coef()</code> (<code>thr</code> , <code>thr_se</code> ; <code>beta</code> is not converted); or a numeric vector of <code>theta</code> (give <code>from</code>). Other columns are kept.
...	One or more links (<code>kits_linking()</code> , <code>kits_link_steps</code>), applied in this order.
from	The scale of x when x does not record it (for example a numeric vector).

Details

Details: `kits_algorithm("wle")` (the conversion) and `tutorial_algorithms/02_scoring.html`.

Value

x on the scale where the last link ends (recorded in `attr(, "scale")`), with the same column names; with draws also `se_link` and `se_total`.

Examples

```
betas <- list(a1 = -2, a2 = -1, a3 = c(-0.5, 0.5), a4 = 0, a5 = c(0.5, 1.5), a6 = 2, a7 = 3)
X03 <- kits_sim(rnorm(500, 0), betas[1:5], seed = 1)
X36 <- kits_sim(rnorm(500, 2), betas[3:7], prefix = "q", seed = 2)
item03 <- kits_item(X03, scale = "lan M03")
item36 <- kits_item(X36, scale = "lan M36")
co <- kits_linking(item36, item03, n_boot = 0)
anchors <- data.frame(item_id = c("a1", "a2", "a3", "a4", "a5"), tau = c(40, 47, 51, 55, 60))
d_link <- kits_linking(item03, anchors, n_boot = 0)

s36 <- kits_subj(X36, item36)

# One link per line: M36 -> M03, then M03 -> D
D36 <- s36 |>
  kits_convert(co) |>
```

```

    kits_convert(d_link)
head(D36)

# The same in one call
all.equal(D36, kits_convert(s36, co, d_link))

# A numeric vector: give its scale
c(-1, 0, 1) |>
  kits_convert(d_link, from = "lan M03")

# Chosen theta values with their linking error (links made with n_boot > 0)
co_boot <- kits_linking(item36, item03, n_boot = 100)
d_boot <- kits_linking(item03, anchors, n_boot = 100)
data.frame(theta = c(-1, 0, 1)) |>
  kits_convert(co_boot, d_boot, from = "lan M36")

# The wrong order stops
try(s36 |> kits_convert(d_link))

```

kits_dif

Differences between groups: LR test and item DIF

Description

Calibrates each group of person-waves on its own by CML and tests whether the step parameters are the same (Andersen, 1973): $LR = 2 (\text{sum of } \log L_g - \log L_{\text{all}})$, on the steps each group observes. It returns each group's steps on a common origin and the Wald z of the difference between every two groups; with large N , read these sizes rather than the p value. With two groups it also gives each item's DIF (the item-pair method of Bechger & Maris, 2015, as dexter's `DIF()`): `dif` is the mean over the item's steps, in logits, positive is harder for the ref group (with `ref = "boy"`, favors girls). `print(d)` shows the LR test.

Usage

```

kits_dif(
  data,
  group,
  item = NULL,
  ref = NULL,
  by = NULL,
  persons = NULL,
  unobserved = c("steps", "drop"),
  ...
)

## S3 method for class 'kits_dif'
print(x, digits = 2, n = NULL, ...)

```

```
## S3 method for class 'kits_dif'
summary(object, digits = 2, n = 10, ...)

## S3 method for class 'kits_dif_summary'
print(x, ...)

## S3 method for class 'kits_dif'
plot(x, type = c("items", "steps"), items = NULL, cut = 0.3, ...)
```

Arguments

data	Responses: a matrix (one row per person-wave, one column per item, row names are person ids), a wide or long data.frame, or a kits_resp() (see kits_item()).
group	The groups: <ul style="list-style-type: none"> • a vector or factor with one group per person-wave: one value per row of data (for a long data.frame, one per response row), or named by visit_id; NA is left out, with a message; • "raw": low and high raw scores, split at the median within each item pattern (the median goes to the low group); • "wave": one group per wave (wave_m of a kits_resp(), or of persons); • the name of another column of a kits_resp() or of persons, for example "sex". Not given: the column by of a kits_resp() .
item	Optional: the kits_item() calibration of the same data, used as the joint model (a calibration of other data stops with an error). By default it is calibrated here.
ref	The reference group of the item DIF (two groups). By default "boy" when the groups are boy and girl, otherwise the first group.
by	With a kits_resp() and no group: the column with the groups, by default "sex".
persons	Optional data.frame with visit_id and the column named by group (for example wave_m), when data is not a kits_resp() with that column.
unobserved	Steps that a group does not observe: "steps" (default) calibrates each group on the steps it observes; "drop" leaves an item that lacks a score category in any group out of every group and of the joint calibration (as eRm's LRtest()).
...	In kits_dif() : not used (an argument the function does not have stops with an error). In print() : passed to print() for a data.frame.
x	A kits_dif (or its summary).
digits	Decimals to print (only the printout is rounded; the object keeps full precision).
n	Rows to print (see print.kits_table()); for summary() , the number of step differences to show (largest difference first).
object	A kits_dif .
type	What plot() draws: "items" (the default with two groups; each item's dif with its 95% CI, beyond +/- cut in colour) or "steps" (the graphical model check, the default with more groups: with two groups the beta of each step in the second group against the first, orange beyond Wald z 1.96; with more groups one panel per group against the joint calibration).

items	Which items to draw (item ids); by default all of them.
cut	For type = "items": the size of DIF highlighted (logits; the cut of <code>kits_dif_combine()</code>).

Details

Details: `kits_algorithm("dif")` and `tutorial_algorithms/05_checks.html`.

Value

A data.frame (class `kits_dif`): with two groups one row per compared item, `item_id` and `dif` (logits; no rows with more groups), and the attributes

- `lrtest`: LR, df, p, split (how the groups were formed), rule (unobserved) and groups, one row per group: group, n, n_info (informative person-waves), n_items, n_steps (steps estimated), n_par (free step parameters), logLik;
- `steps`: one row per group and estimated step: group, item_id, item_score, beta, se (on the origin of the joint calibration);
- `full`: the joint calibration: item_id, item_score, beta, se, with attributes logLik and n_par;
- `diff`: one row per pair of groups and step: item_id, item_score, group_1, group_2, diff (beta in group_2 minus beta in group_1), se, z;
- `unobserved`: the steps of answered items that a group does not estimate: group, item_id, item_score, reason, left_out (the item's responses in that group are left out of the joint calibration too);
- `with two groups` (otherwise NULL): overall (stat, df, p: the test that any item has DIF), groups (ref, focal and the person-waves n of each), se (of each dif) and left_out (items not compared: in one group only, or with different scores);
- `diagnostics` (see `kits_check()`) and `info` (see `kits_info()`).

References

- Andersen, E. B. (1973). A goodness of fit test for the Rasch model. *Psychometrika*, 38, 123-140. doi:10.1007/BF02291180
- Bechger, T. M., & Maris, G. (2015). A statistical test for differential item pair functioning. *Psychometrika*, 80(2), 317-340. doi:10.1007/s113360149408y

See Also

`kits_dif_combine()` to combine two cohorts, `kits_drift()`, `kits_itemfit()`.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5, a5 = 0)
theta <- seq(-3, 3, length.out = 300)
boys <- kits_sim(theta, betas, prefix = "b", seed = 1)
girls <- kits_sim(theta, modifyList(betas, list(a5 = 1)), prefix = "g", seed = 2)
X <- rbind(boys, girls)
```

```
# a5 is harder for girls: negative, favors boys
dif <- X |> kits_dif(rep(c("boy", "girl"), each = 300))
dif
attr(dif, "lrtest")$p

# Andersen's median split; do the steps stay the same across waves?
kits_dif(X, "raw")
resp <- kitm03$responses |>
  dplyr::filter(domain == "lan", wave_m <= 12) |>
  kits_resp()
kits_dif(resp, "wave")

# Each item with its 95% CI; beyond +/- cut in colour; then each step in both groups
p <- plot(dif)
p <- plot(dif, type = "steps")
if (interactive()) p
```

kits_dif_combine

Combined DIF decision over two cohorts

Description

Decides the direction of each item in each cohort (beyond cut favors the focal group, below -cut the reference group, else "none") and combines the two: an item in one cohort only takes that cohort's result, opposite signs with one side beyond cut are "inconsistent", otherwise the direction that exceeds cut. Both `kits_dif()` results must use the same ref.

Usage

```
kits_dif_combine(d1, d2, cut = 0.3, cohorts = NULL)
```

Arguments

d1, d2	The <code>kits_dif()</code> results of the two cohorts (plain tables are read as ref = "boy").
cut	The cutoff for the decision (logits).
cohorts	The names of the two cohorts, used in the output column names (dif_<name>, dir_<name>); by default the names of the two arguments, otherwise "1" and "2".

Details

Details: `kits_algorithm("dif")` and `tutorial_algorithms/05_checks.html`.

Value

A data.frame, one row per item: item_id, dif_ and dir_ for both cohorts, and the combined DIF ("none", "favors girls", "favors boys" or "inconsistent" with ref = "boy"), which `kits_item_table()` uses to colour the items.

Examples

```
d1 <- data.frame(item_id = c("a1", "a2", "a3", "a4"), dif = c(0.1, 0.5, 0.4, -0.5))
d2 <- data.frame(item_id = c("a1", "a2", "a3", "a5"), dif = c(0.2, 0.1, -0.2, 0.6))
kits_dif_combine(d1, d2)
```

kits_drift

Item drift: do the items behave the same in a new wave of data?

Description

Calibrates the new data freely ([kits_item\(\)](#)), links its step thresholds to those of ref ([kits_linking\(\)](#): a mean/sigma line with one round of screening), and flags the items with a dropped step as drifted. Steps need at least min_n persons below and above to be judged. Items that did not drift, or carry too little information to judge, are suggested for fixing (drift\$fix), ready for kits_item(X_new, fixed_params = drift\$fix) on the scale of ref. An item with a higher score in the new data than in ref ("score range differs") is not judged, since its thresholds depend on all its steps; it is suggested for fixing, and [kits_item\(\)](#) then keeps its old steps and estimates the new one. Details: vignette("kits-advanced", package = "kits"), sections 8.1 and 8.2.

Usage

```
kits_drift(data, ref, cut = 2, min_n = 30, ...)

## S3 method for class 'kits_drift'
print(x, digits = 2, ...)

## S3 method for class 'kits_drift'
plot(x, type = c("scatter", "difference"), items = NULL, ...)
```

Arguments

data	New data: a matrix (one row per person-wave, one column per item, row names are person ids), a wide data.frame of the same shape, or a long data.frame with visit_id, item_id, item_score (see kits_item()). Items with a score gap are not judged.
ref	The kits_item() result of the earlier calibration, or its thresholds table, or a table of step parameters (item_id, item_score, beta; the thresholds are computed).
cut	The cutoff for dropping, in residual SDs.
min_n	The minimum number of persons needed both below and above a step.
...	Not used.
x	A kits_drift.
digits	Decimals to print (only the printout is rounded; the object keeps full precision).
type	What plot() draws: "scatter" (the default; the new threshold of each step against the old one, with the line of the judged steps) or "difference" (the distance of each step from that line).
items	Which items to draw (item ids); by default all of them.

Value

A `kits_drift`: a list that prints briefly. Its elements:

- `fix`: the ref parameters (`item_id`, `item_score`, `beta`) of the items suggested for fixing, with `attr(, "scale")`; NULL when ref has no beta column.
- `intercept`, `slope`: the line from the free calibration of the new data to the ref scale.
- `points`: one row per step: the thresholds, the numbers below and above, judged, the residual and whether it was dropped.
- `items`: one row per item: `item_id`, `max_abs_resid`, `drift` (NA when not judged), `suggest_fix`, `reason` (why the item is not judged).
- `steps_free`: the step table of the free calibration of the new data.
- `scale`, `diagnostics` (see [kits_check\(\)](#)), `info` (see [kits_info\(\)](#)).

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5, a5 = c(-0.5, 0.5), a6 = 1.5)
old <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
item_old <- kits_item(old)

drifted <- modifyList(betas, list(a3 = 2)) # a3 became harder in the new data
new <- kits_sim(seq(-2, 4, length.out = 400), drifted, prefix = "n", seed = 2)
drift <- kits_drift(new, item_old)
drift$items

# Calibrate the new data, fixing the items suggested for fixing
item_new <- kits_item(new, fixed_params = drift$fix)
coef(item_new)

# New against old thresholds; the steps of the drifted item in orange
p <- plot(drift)
p <- plot(drift, type = "difference")
if (interactive()) p
```

kits_esf

Elementary symmetric functions of CML and the conditional item-score distributions

Description

A teaching view of the core of conditional maximum likelihood (CML): for one set of items (one item pattern), the elementary symmetric functions `gamma_r` and the distribution of each item score given the raw score `r` that they imply, with its mean and variance. The numbers come from the same internal code as [kits_item\(\)](#), [kits_itemfit\(\)](#) and [kits_item_restscore\(\)](#).

Usage

```

kits_esf(item, items = NULL, pattern_id = NULL, ...)

## S3 method for class 'kits_esf'
print(x, ...)

## S3 method for class 'kits_esf'
as.data.frame(x, ...)

## S3 method for class 'kits_esf'
plot(x, items = NULL, ...)

```

Arguments

item	The item parameters: a kits_item() result (or its fit, a kits() , or a data.frame with item_id, item_score, beta).
items	Optional item ids: the items of the pattern (default: all items). For plot(): which items to draw (default: all items of x).
pattern_id	Optional: one item pattern of item\$design (a kits_item() or kits_cml() result); its items. Give items or pattern_id, not both.
...	Not used: an argument the function does not have stops with an error (class kits_error_input).
x	A kits_esf object.

Details

Details: the models tutorial (tutorial_models/kits_models.html), sections 2 and 6.

Value

A kits_esf object, a list with

- gamma: one row per raw score: r, gamma, log_gamma (also as.data.frame(x));
- prob: the conditional distributions, long: item_id, r, x, prob ($P(X_i = x | r)$; 0 where x cannot occur given r);
- moments: item_id, r, expected ($E(X_i | r)$), variance ($Var(X_i | r)$);
- items, max_score (the highest raw score), steps (the betas used), scale, info.

See Also

[kits_itemfit\(\)](#) (conditional infit and outfit), [kits_item_restscore\(\)](#), [kits_algorithm\(\)](#) (the topic "cml").

Examples

```
# Two items by hand: a1 with 2 steps, a2 with 1 step
parms <- data.frame(item_id = c("a1", "a1", "a2"), item_score = c(1, 2, 1),
                    beta = c(-1, 0.5, 0.3))

esf <- kits_esf(parms)
esf
esf$prob

# The language calibration of kitm03; bk03 is the form of 12 to 24 months
item <- kitm03$responses |>
  dplyr::filter(domain == "lan") |>
  kits_resp() |>
  kits_item(scale = "lan M03")
esf <- kits_esf(item, pattern_id = "bk03")
esf
p <- plot(esf)
if (interactive()) p
```

kits_info

Provenance of a result

Description

Every main result of kits records how it was made: the call (object names only; data given by value show as <data>), the kits version, the settings, and a fingerprint of the data (counts, item ids, md5 hashes of the sorted ids and responses). The record holds no data and no time stamp, so a result can be shared and the same input gives an identical object.

Usage

```
kits_info(x)

## S3 method for class 'kits_info'
print(x, ...)
```

Arguments

x	A result of kits, for example of <code>kits_item()</code> , <code>kits_subj()</code> , <code>kits_linking()</code> or <code>kits_norms()</code> .
...	Not used.

Value

A kits_info list (printed briefly): call, version, settings, data, and, when they apply:

- reference: the fixed parameters; item_data: the data the item parameters came from;
- persons: a fingerprint of the person information (persons of `kits_subj()`, or wave_m);
- input: the name and form of a data.frame or `kits_subj()` given as data;
- conversions (links applied by `kits_convert()`) and links (their intercept, slope).

Examples

```

betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
item <- kits_item(X, scale = "demo")
kits_info(item)
identical(kits_item(X, scale = "demo"), item)

# Person information given to kits_subj() is fingerprinted too
s <- kits_subj(X, item, persons = data.frame(visit_id = rownames(X), wave_m = 12))
kits_info(s)

```

kits_information	<i>Test information and expected score as functions of theta</i>
------------------	--

Description

Each returns a function of theta for a set of items: `kits_information()` the test information $I(\theta) = \sum_i \text{Var}(X_i | \theta)$ under the PCM, and `kits_expected_score()` the expected raw score $\sum_i E(X_i | \theta)$ (the test characteristic curve). theta is on the calibration scale of the step parameters (beta).

Usage

```

kits_information(item, ...)

## Default S3 method:
kits_information(item, items = NULL, pattern_id = NULL, ...)

## S3 method for class 'kits_item'
kits_information(item, items = NULL, pattern_id = NULL, ...)

## S3 method for class 'kits'
kits_information(item, ...)

## S3 method for class 'kits_fit'
kits_information(item, items = NULL, pattern_id = NULL, ...)

kits_expected_score(item, ...)

## Default S3 method:
kits_expected_score(item, items = NULL, pattern_id = NULL, ...)

## S3 method for class 'kits_item'
kits_expected_score(item, items = NULL, pattern_id = NULL, ...)

## S3 method for class 'kits'
kits_expected_score(item, ...)

```

```
## S3 method for class 'kits_fit'
kits_expected_score(item, items = NULL, pattern_id = NULL, ...)

## S3 method for class 'kits_information'
print(x, ...)

## S3 method for class 'kits_expected_score'
print(x, ...)
```

Arguments

item	The item parameters: a <code>kits_item()</code> result, its fit (<code>kits_cml()</code>), a <code>kits()</code> , or a data.frame with item_id, item_score, beta.
...	Not used.
items	Optional item ids: the items of the test (default: all items).
pattern_id	Optional: one item pattern of item\$design (a <code>kits_item()</code> or a <code>kits_cml()</code> result); its items.
x	A kits_information or kits_expected_score function.

Details

Details: `kits_algorithm("information")` and `tutorial_algorithms/06_scale.html`; the models tutorial (`tutorial_models/kits_models.html`), section 1.

Value

A function of theta (a numeric vector), of class `kits_information` or `kits_expected_score`, that returns one value per theta. It keeps `attr(, "parms")` (the betas), `attr(, "items")` and `attr(, "scale")`.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
item <- kits_item(X)
info <- kits_information(item)
info
info(c(-2, 0, 2))
1 / sqrt(info(0)) # the SE of theta at 0

expected <- kits_expected_score(item, items = c("a1", "a2"))
expected(c(-2, 0, 2))
```

kits_item	<i>Item calibration: the PCM by CML in RTMB, plus the threshold of every step</i>
-----------	---

Description

Calibrates the items with the partial credit model by conditional maximum likelihood, then computes the threshold `thr` of every step: the ability at which $P(X \geq k) = .5$ (RP50). The origin is the sum of all $\beta = 0$, or the fixed items when `fixed_params` is given. The calibration defines a scale, labelled by `scale`; the label travels with the results, so that a wrong link stops.

Usage

```
kits_item(data, ..., scale = NULL)

## S3 method for class 'kits_item'
print(x, digits = 2, n = 6, ...)

## S3 method for class 'kits_item'
coef(object, rp = 0.5, ...)

## S3 method for class 'kits_item'
plot(
  x,
  type = c("category", "expected", "wright", "fit", "observed"),
  items = NULL,
  theta = NULL,
  scores = NULL,
  data = NULL,
  group = NULL,
  step = 1,
  persons = NULL,
  ...
)
```

Arguments

<code>data</code>	For <code>type = "observed"</code> : the responses (as in <code>kits_item()</code>).
<code>...</code>	Only <code>fixed_params</code> : items to fix at earlier estimates, for example <code>fixed_params = dplyr::filter(coef(old), item_id %in% fix)</code> (a whole <code>kits_item</code> is also accepted). Steps are fixed one by one: a fixed item with a higher score in <code>data</code> than in <code>fixed_params</code> keeps its fixed steps (their β), and the new steps are estimated (a new wave where older children reach a score no younger child had); such an item needs no 0 score, but its scores above the fixed steps must be consecutive. The thresholds <code>thr</code> of the item change, as they depend on all its steps.

scale	A label for the scale this calibration defines, for example "1an M03". Always give it; by default the domain and cohort of a <code>kits_resp()</code> , the name of data, or "scale 1". With <code>fixed_params</code> that record a scale, that scale is used (another scale stops).
x, object	A <code>kits_item</code> .
digits	Decimals to print (only the printout is rounded; the object keeps full precision).
n	Steps to print (the first 6 by default; Inf for all). Fixed steps are marked in a fixed column.
rp	The response probability of the thresholds: thr of step k is the theta at which $P(X \geq k) = rp$. Default 0.5 (RP50, as calibrated); links use RP50 only.
type	What <code>plot()</code> draws: "category" (the default; the probability of each score against theta, one panel per item), "expected" (the expected score of each item), "wright" (a Wright map; needs scores, on the scale of the thresholds), "fit" (<code>x\$itemfit</code>) or "observed" (needs data: the share of children who pass each item against their theta, one line per group, in bins of 0.5 logits with at least 10 children; the ticks at the top are the thetas of those who pass, at the bottom of those who fail; grey: the model; bottom right: infit and outfit of <code>kits_itemfit()</code> on data).
items	Which items to draw (item ids); by default all of them, or the first 9 when there are more (with a message).
theta	The grid of theta (on the calibration scale); by default the range of the thresholds plus 3 logits on each side.
scores	For type = "wright": the scores of the children (<code>kits_subj()</code> , or a <code>data.frame</code> with theta). For type = "observed": optional, a <code>data.frame</code> with <code>visit_id</code> and theta on the scale of the betas; by default <code>kits_subj(data, x)</code> .
group	For type = "observed": the groups, one line each, as in <code>kits_dif()</code> (for example "wave", "sex" or one value per row of data); by default one line.
step	For type = "observed": pass is <code>item_score >= step</code> , as in <code>kits_item_table()</code> (one number, or numbers named by item).
persons	For type = "observed": optional <code>data.frame</code> with <code>visit_id</code> and the column named by group, as in <code>kits_dif()</code> .

Details

Details: `kits_algorithm("cml")` and `tutorial_algorithms/01_calibration.html`; the models tutorial (`tutorial_models/kits_models.html`), sections 2, 3 and 5.

Value

A `kits_item`: a list that prints briefly. Its elements:

- thresholds: one row per step: `item_id`, `item_score` (the step), `beta`, `beta_se`, `thr`, `thr_se`. Also returned by `coef()`.
- fit: the CML fit (class `kits_fit`, see `kits_cml()`).
- stats: the sufficient statistics of the CML (class `kits_stats`, see `kits_stats()`).

- `n_persons`: the number of person-waves in the calibration.
- `left_out`: fixed items left out of the calibration (no 0 score, or one score only, and no score above their fixed steps).
- `design`: the item patterns (`pattern_id`, `item_id`), including left-out items.
- `scale`: the scale label (also `attr(coef(x), "scale")`).
- `itemfit`: the conditional item fit on the calibration data, as `kits_itemfit()` (NULL when it cannot be computed).
- `diagnostics`: the checks of the calibration (see `kits_check()`).
- `info`: the call, settings and data fingerprint (see `kits_info()`).

Examples

```

betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
item <- kits_item(X, scale = "lan M03")
item
coef(item)

# A new wave: fix the items of the first calibration (the new scale is that of the fixed items)
betas_new <- c(betas, list(a5 = c(0.5, 1.5)))
X_new <- kits_sim(seq(-2, 4, length.out = 400), betas_new, prefix = "n", seed = 2)
item_new <- kits_item(X_new, fixed_params = coef(item))
print(item_new, n = 8)

# One wave at a time: old betas never move and earlier scores are never estimated again.
# In wave 2, a2 drifts (it gets a new version, a2_w2), a3 reaches a score of 2 (step 1 stays
# fixed, step 2 is estimated; the thresholds of a3 change) and a5 is new.
betas_1 <- c(betas, list(a6 = c(-1, 1), a7 = 1))
X_1 <- kits_sim(seq(-3, 3, length.out = 400), betas_1, seed = 1)
bank <- coef(kits_item(X_1, scale = "lan M03"))[c("item_id", "item_score", "beta", "beta_se")]
betas_2 <- c(betas_1[c("a1", "a4", "a6")],
             list(a2 = c(1, 2), a7 = 1, a3 = c(0.5, 2), a5 = c(1, 2)))
X_2 <- kits_sim(seq(-1, 5, length.out = 600), betas_2, prefix = "m", seed = 3)
drift <- kits_drift(X_2, bank) # stop here when too many common items drift
moved <- drift$items$item_id[drift$items$drift %in% TRUE]
colnames(X_2)[colnames(X_2) %in% moved] <- paste0(moved, "_w2")
item_2 <- kits_item(X_2, fixed_params = drift$fix)
added <- dplyr::anti_join(coef(item_2), bank, by = c("item_id", "item_score"))
bank <- rbind(bank, added[names(bank)])
bank

# Thresholds at RP80: the theta at which P(X >= k) = .8
coef(item, rp = 0.8)

# The probability of each score of each item
p <- plot(item)
p <- plot(item, items = c("a1", "a2"))
# The expected score of each item
p <- plot(item, type = "expected")
# A Wright map: the children beside the thresholds

```

```

p <- plot(item, type = "wright", scores = kits_subj(X, item))
# Item fit: infit and outfit of each item (x$itemfit)
p <- plot(item, type = "fit")
# Observed share passing against theta, by a group, beside the model (grey)
p <- plot(item, type = "observed", data = X, group = rep(c("a", "b"), length.out = nrow(X)))
if (interactive()) p

```

kits_item_restscore *Item-restscore association: observed against expected gamma*

Description

Kreiner's (2011) test of item fit: for each item, the Goodman-Kruskal gamma of item score x rest score (the sum of the other items) is compared with the gamma the PCM implies given the item parameters. Each row is one item in one item pattern with at least min_n persons. gamma_obs below gamma_exp means the item separates persons less than the model implies, above means more (for example local dependence); with large N, look at the size of the difference.

Usage

```

kits_item_restscore(data, item, min_n = 100, ...)

## S3 method for class 'kits_item_restscore'
print(x, digits = 3, n = 10, ...)

## S3 method for class 'kits_item_restscore'
plot(x, items = NULL, ...)

```

Arguments

data	Responses, as in <code>kits_item()</code> : a matrix, a wide or long data.frame, or a <code>kits_resp()</code> table (its columns of the child are then the persons). One domain only; gaps in the observed scores are allowed.
item	The item parameters: the result of <code>kits_item()</code> (or its fit), or a <code>kits()</code> , or a data.frame with item_id, item_score, beta.
min_n	Smallest number of persons in an item pattern for it to be tested (skipped patterns get a note in the diagnostics).
...	Not used: an argument the function does not have stops with an error (class <code>kits_error_input</code>).
x	A <code>kits_item_restscore</code> .
digits	Decimals to print.
n	Rows to print in each of the two lists (significant rows; largest differences).
items	Which items to draw (item ids); by default all of them.

Details

Details: the models tutorial (tutorial_models/kits_models.html), sections 6 and 9.2.

Value

A data.frame (class kits_item_restscore) with one row per item and item pattern:

- item_id, pattern_id (as in `kits_subj()`), n (persons);
- gamma_obs, gamma_exp: the observed gamma and the gamma expected under the PCM;
- se (asymptotic SE of gamma_obs; Goodman & Kruskal, 1972), z, p (two-sided, normal);
- p_adj: Benjamini-Hochberg (1995) over all rows.

NA when the gamma cannot be computed (the item or the rest score is constant in the pattern). The skipped patterns are in `attr(x, "skipped")`.

References

Kreiner, S. (2011). A note on item-restscore association in Rasch models. *Applied Psychological Measurement*, 35, 557-561.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society B*, 57, 289-300.

Goodman, L. A., & Kruskal, W. H. (1972). Measures of association for cross classifications, IV: Simplification of asymptotic variances. *Journal of the American Statistical Association*, 67, 415-421.

See Also

`kits_itemfit()` for infit, outfit and the item-rest correlation.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5, a5 = 0)
X <- kits_sim(seq(-3, 3, length.out = 500), betas, seed = 1)
X[, "a5"] <- rep(0:1, 250) # a5 is unrelated to the others
item <- kits_item(X)
rs <- kits_item_restscore(X, item)
rs

# Observed against expected gamma; off the line = misfit
p <- plot(rs)
if (interactive()) p
```

kits_item_steps

The calibration step by step: counts, CML, thresholds

Description

`kits_item()` calibrates in three steps, and each step is also a function: `kits_stats()` counts the sufficient statistics (n_ig, persons with a score of at least g on item i; n_br, persons with raw score r in item pattern b), `kits_cml()` maximizes the conditional likelihood of the PCM from these counts, and `kits_thresholds()` gives the threshold of every step with its SE. They give the numbers of `item$stats`, `item$fit` and `coef(item)`.

Usage

```

kits_stats(data, fixed_params = NULL, ...)

## S3 method for class 'kits_stats'
print(x, n = 6, ...)

kits_cml(stats)

kits_thresholds(fit)

```

Arguments

<code>data</code>	For type = "observed": the responses (as in kits_item()).
<code>fixed_params</code>	Optional fixed step parameters (item_id, item_score, beta), as in kits_item() . Fixed items with no 0 score or one score are left out (left_out).
<code>...</code>	Not used (an argument the function does not have stops with an error).
<code>x</code>	A kits_stats.
<code>n</code>	Rows of each table to print (Inf for all).
<code>stats</code>	A kits_stats() result (or item\$stats of a kits_item()).
<code>fit</code>	A kits_cml() result (or item\$fit).

Details

Details: [kits_algorithm\("cml"\)](#) and [tutorial_algorithms/01_calibration.html](#); the models tutorial ([tutorial_models/kits_models.html](#)), sections 2, 3 and 5.

Value

- `kits_stats()`: a kits_stats, a list that prints briefly. Its elements: items (the items, in the order of `sort()`); steps (one row per step, item_id, item_score (the step g) and n (n_ig)); item patterns (one row per item pattern, pattern_id, items (a list of item ids), n (persons) and max_raw); raw (one row per item pattern and raw score 0 to max_raw, pattern_id, raw and n (n_br)); fixed (the fixed parameters of the items in the counts, or NULL) and left_out (the fixed parameters of the fixed items left out, or NULL); n_persons (the number of person-waves) and info (see [kits_info\(\)](#)).
- `kits_cml()`: a kits_fit, with coef (item_id, item_score, beta, SE_beta), est (beta, acov.beta, nll), inputs (design, method = "CML", n_persons, left_out) and optim (the convergence record).
- `kits_thresholds()`: a table with one row per step: item_id, item_score, beta, beta_se, thr, thr_se.

See Also

[kits_item\(\)](#) (all three steps in one call), [kits_algorithm\(\)](#).

Examples

```

betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
X[1:100, "a4"] <- NA # two item patterns

# 1 the sufficient statistics
stats <- kits_stats(X)
stats
stats$steps

# 2 the CML
fit <- kits_cml(stats)
fit
coef(fit)

# 3 the thresholds
thresholds <- kits_thresholds(fit)
thresholds

# The same numbers are in the calibration
identical(stats$steps, kits_item(X)$stats$steps)

```

kits_item_table	<i>Item table: each item with its D, ages, DIF and stem, for the report plots</i>
-----------------	---

Description

One row per item: the D of its milestone step, the age at which a share of the children pass it, the age at which the norm median reaches its D, its sex DIF and its stem. The table keeps the norm (summary numbers, no child's data), so `plot()` draws the growth curves with the items (`type = "growth"`, one panel per domain) or the developmental ladder (`type = "ladder"`).

Usage

```

kits_item_table(
  thr,
  norms = NULL,
  data = NULL,
  dif = NULL,
  stems = NULL,
  rp = 0.5,
  dif_cut = 0.3,
  step = 1
)

## S3 method for class 'kits_item_table'
plot(

```

```

x,
type = c("growth", "ladder"),
items = NULL,
ages = c(3, 6, 9, 12, 18, 24, 36, 48, 60, 72, 84),
age_ticks = AGE_TICKS,
y_range = NULL,
...
)

```

Arguments

thr	The thresholds, on the scale of the norm: a <code>kits_item()</code> converted with <code>kits_convert()</code> , or its thresholds table (<code>coef()</code> , <code>item_id</code> , <code>item_score</code> , <code>thr</code> , optionally <code>domain</code> and <code>column</code>), or a list of them (stacked in order).
norms	Optional: the norm, a <code>kits_norms()</code> result for one domain, or by domain (by = "domain"); P10, P50 and P90 are used. Without it the table has D and, with data, the observed pass ages, but no place on the growth chart (<code>age_reach</code> , ...) and no <code>plot()</code> .
data	Optional responses with ages: a <code>kits_resp()</code> , or what it takes (a long data.frame, or a wide data.frame of item scores), with <code>age_m</code> and <code>wave_m</code> . For each item, a logistic regression of passing (a score of at least <code>step</code>) on age gives the age at which a share <code>rp</code> pass (NA when the pass rate of the waves does not cross <code>rp</code>).
dif	Optional: a <code>kits_dif_combine()</code> or a <code>kits_dif()</code> result.
stems	Optional: the item stems, a data.frame with <code>item_id</code> and <code>stem</code> , or a character vector named by item.
rp	The response probability of <code>thr</code> and of the ages (default 0.5). A <code>kits_item()</code> gives its thresholds at this RP (<code>coef(item, rp = rp)</code>); a table must hold thresholds at this RP (its <code>attr(, "rp")</code> , RP50 when not recorded), otherwise the function stops.
dif_cut	With one <code>kits_dif()</code> as <code>dif</code> : the size of DIF (logits) beyond which an item favors a group (default 0.3, as in <code>kits_dif_combine()</code>). Not used for a <code>kits_dif_combine()</code> result, whose DIF column is already decided.
step	The pass step: a score of at least <code>step</code> passes, and D is the threshold of that step. One whole number for every item (default 1, the milestone step), or a vector named by item (items not named use 1), e.g. <code>c(cogD01 = 3)</code> for a reversed 0-4 item whose pass is 3 or more. Items whose steps differ are not the same kind of milestone; the <code>step</code> column says which.
x	A <code>kits_item_table</code> .
type	"growth" (the default; growth curves with the items) or "ladder" (the developmental ladder).
items	Which items to draw (item ids); by default all of them.
ages	For type = "ladder": the ages of the ticks on the right.
age_ticks	For type = "growth": the age ticks.
y_range	For type = "growth": the range of the D axis, the same in every panel; by default from the data.
...	Not used.

Details

Details: vignette("kits-advanced", package = "kits"), sections 7.3 to 7.5 and 10.9.

Value

A `kits_item_table`, a data.frame with one row per item; `attr(, "norm")` keeps the norm, `attr(, "rp")` the RP; with data, the diagnostics list the items with perfect separation by age (their age is only known between the oldest child who fails and the youngest who passes). Its columns:

- D: the threshold of the pass step (`item_score == step`), on the scale of the norm.
- step: the pass step, only when step is not 1 for every item.
- domain, column: the domain of the item and its column of the ladder (default the domain).
- age50 (age75, ... for another rp): the age at which a share rp pass, from data.
- DIF: "none", "favors girls", "favors boys" or "inconsistent", from dif.
- stem: the item stem (any language), from stems; the item id when not given.
- age_reach, below_range, above_range, empirical, x: the place on the growth chart (the empirical age when there is one, else the age at which the norm median reaches D); only with norms.

Examples

```
set.seed(1)
age <- runif(800, 3, 24)
D <- 20 + 15 * log(age) + rnorm(800, 0, 3)
norm <- kits_norms(D, age, ages = 3:24, degrees = c(3, 2, 1, 1),
  by = data.frame(domain = rep("lan", 800)))
thr <- data.frame(item_id = c("a1", "a2", "a3", "a4"), item_score = 1, thr = c(40, 55, 65, 60),
  domain = "lan")
pass <- sapply(c(40, 55, 65, 60), \(b) rbinom(800, 1, plogis((D - b) / 3)))
colnames(pass) <- thr$item_id
resp <- data.frame(visit_id = 1:800, age_m = age, wave_m = round(age), pass)
dif <- data.frame(item_id = c("a2", "a3", "a4"),
  DIF = c("favors girls", "favors boys", "inconsistent"))
stems <- c(a1 = "Lifts head", a2 = "Says two words", a3 = "Names colours",
  a4 = "Points to pictures")
item_tab <- kits_item_table(thr, norm, data = resp, dif = dif, stems = stems)
item_tab

# A 0-4 item whose pass is a score of 3 or more
thr4 <- rbind(thr, data.frame(item_id = "a5", item_score = 1:4, thr = c(30, 45, 58, 70),
  domain = "lan"))
kits_item_table(thr4, norm, step = c(a5 = 3))

# No norm: D and the observed pass ages from the responses only
kits_item_table(thr, data = resp)
p <- plot(item_tab)
p <- plot(item_tab, type = "ladder", ages = c(6, 12, 18))
if (interactive()) p
```

kits_itemfit

Item fit: infit and outfit mean squares, and the item-rest correlation

Description

For each item, the infit and outfit mean squares of the residuals under the PCM (expected value 1; above 1 more random, below 1 more predictable than the model expects) and the item-rest correlation *r_{ir}* (weighted mean over item patterns). By default the residuals are conditional on the raw score within the item pattern (Christensen, Kreiner & Mesbah, 2013), so no ability estimate is needed; persons with a zero or a perfect score are left out.

Usage

```
kits_itemfit(data, item, type = c("conditional", "wle"), ...)

## S3 method for class 'kits_itemfit'
plot(
  x,
  type = c("both", "infit", "outfit"),
  items = NULL,
  range = c(0.6, 1.4),
  ...
)
```

Arguments

data	Responses, as in kits_item() : a matrix, a wide or long data.frame, or a kits_resp() table (its columns of the child are then the persons). One domain only; gaps in the observed scores are allowed.
item	The item parameters: the result of kits_item() (or its fit), or a kits() , or a data.frame with <i>item_id</i> , <i>item_score</i> , <i>beta</i> .
type	What <code>plot()</code> draws: "both" (the default; infit and outfit), "infit" or "outfit".
...	Not used: an argument the function does not have stops with an error (class <code>kits_error_input</code>).
x	A <code>kits_itemfit</code> .
items	Which items to draw (item ids); by default all of them.
range	The acceptable range of the mean squares, shaded (default 0.6 to 1.4); the points are coloured by the flag against it (circles infit, squares outfit).

Details

Details: [tutorial_algorithms/05_checks.html](#).

Value

A data.frame (class kits_itemfit) with one row per item: item_id, n (responses used), pvalue (the mean score as a share of the maximum), rir, infit, outfit, and flag, against the range 0.6 to 1.4 (Wright & Linacre, 1994):

- "underfit": infit or outfit above 1.4 (it wins when an item is also below 0.6).
- "overfit": otherwise, infit or outfit below 0.6.
- "ok": both within 0.6 to 1.4, the limits included; NA when infit and outfit are NA.

References

Christensen, K. B., Kreiner, S., & Mesbah, M. (Eds.). (2013). *Rasch models in health*. ISTE and Wiley.

Wright, B. D., & Linacre, J. M. (1994). Reasonable mean-square fit values. *Rasch Measurement Transactions*, 8(3), 370.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5, a5 = 0)
X <- kits_sim(seq(-3, 3, length.out = 400), betas, seed = 1)
X[, "a5"] <- sample(0:1, 400, replace = TRUE) # a5 is pure noise
item <- kits_item(X)
fit <- kits_itemfit(X, item)
fit
identical(fit$infit, item$itemfit$infit) # kits_item() keeps the same table

# At each person's WLE: lower on short forms
kits_itemfit(X, item, type = "wle")

# Infit and outfit of each item, the acceptable range shaded
p <- plot(fit)
p <- plot(fit, type = "outfit", range = c(0.5, 1.5))
if (interactive()) p
```

kits_link_check

Independent checks of a link between two calibrations

Description

Estimates a cohort link in two ways that do not use the screened common items, and compares the converted values at chosen theta with the main link (Kolen & Brennan, 2014): same-age groups at the shared waves (a latent normal per cohort and wave, item parameters fixed), and a concurrent CML calibration of the stacked data of both cohorts. Details: the models tutorial (tutorial_models/kits_models.html), section 7, and vignette("kits-advanced", package = "kits"), section 10.7.

Usage

```

kits_link_check(
  link,
  item_from,
  item_to,
  X_from,
  X_to,
  persons = NULL,
  theta = NULL,
  tol = 0.2,
  persons_from = NULL,
  persons_to = NULL
)

## S3 method for class 'kits_link_check'
print(x, digits = 2, ...)

```

Arguments

link	The link to check: a <code>kits_linking()</code> between two calibrations (not to anchors, not with domains).
item_from, item_to	The two separate calibrations (<code>kits_item()</code>) that the link joins, on the scales <code>link\$from_scale</code> and <code>link\$to_scale</code> .
X_from, X_to	The responses of each calibration (a matrix or data.frame, as for <code>kits_item()</code>). The person ids of the two may overlap; they are kept apart in the concurrent calibration.
persons	Optional person information with <code>visit_id</code> and <code>wave_m</code> for both cohorts (ids must not repeat). Without it, or without shared waves, the same-age check is skipped.
theta	Values on the from scale at which to compare the lines. By default the median theta of each wave of from (or the quartiles without persons).
tol	Tolerance for the difference when the link has no bootstrap draws (units of the target scale; default 0.2).
persons_from, persons_to	The person information of each cohort on its own (<code>visit_id</code> , <code>wave_m</code>), for when the person ids overlap. Give either persons or both of these.
x	A <code>kits_link_check</code> .
digits	Decimals to print.
...	Not used.

Value

A `kits_link_check`: a list with

- methods: one row per method: method, intercept, slope, basis (what it rests on).

- at: one row per method and theta: theta, method, value, diff, se_method, and with draws se_link, z and z_both.
- waves: one row per shared wave (NULL when none): n, latent mean, sd and their SEs of each cohort (_from, _to), and converged.
- concurrent: the concurrent `kits_item()`; lines: the lines of the separate calibrations on its scale.
- from_scale, to_scale, settings (tol), diagnostics (see `kits_check()`) and info (see `kits_info()`).

References

Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd ed.). Springer. doi:10.1007/9781493903177

Examples

```
# Two cohorts measured at 36 months; items a4 to a7 are common
betas <- lapply(seq(-3, 3, length.out = 10), function(b) b + c(-0.5, 0.5))
names(betas) <- paste0("a", 1:10)
set.seed(1)
wave <- rep(c(24, 36, 36, 48), each = 400)
theta <- 2 * log(wave / 24) + rnorm(1600)
young <- kits_sim(theta[1:800], betas[1:7], prefix = "y", seed = 1)
old <- kits_sim(theta[801:1600], betas[4:10], prefix = "o", seed = 2)
persons <- data.frame(visit_id = c(rownames(young), rownames(old)), wave_m = wave)
item_young <- kits_item(young, scale = "young")
item_old <- kits_item(old, scale = "old")
link <- kits_linking(item_old, item_young, n_boot = 100)
chk <- kits_link_check(link, item_old, item_young, old, young, persons = persons)
chk
```

kits_link_steps

A link step by step: points, screening, line, linking error

Description

The four steps of `kits_linking()` as functions of their own: `kits_link_points()` (the points), `kits_link_screen()` (one round of screening), `kits_link_line()` (the mean/sigma line) and `kits_link_error()` (the item-resampling linking error). Piped in this order they give the same link as `kits_linking(..., n_boot = )`. Details: `kits_algorithm("link")` and `tutorial_algorithms/03_linking.html`.

Usage

```
kits_link_points(from, to, step = NULL, to_scale = NULL)
```

```
kits_link_screen(
  points,
  cut = 2,
```

```

    spread = c("sd", "mad", "se"),
    unit = c("step", "item"),
    from_scale = NULL,
    to_scale = NULL
  )

  kits_link_line(screened)

  kits_link_error(link, n_boot = 200, seed = 1)

```

Arguments

from	The calibration to be converted: a <code>kits_item()</code> result (for example the M36 cohort), a <code>kits()</code> , its thresholds table (optionally with a domain column), or a named list of them, one per domain.
to	The target: a calibration (<code>kits_item()</code> , <code>kits()</code> or a thresholds table), or a data.frame of anchor items with <code>item_id</code> , <code>tau</code> and optionally <code>step</code> (the pass step).
step	Which step of each item gives the points. Between two calibrations: NULL (every common step) or one whole number. To anchor items: the pass step, "pass" is a score of at least step; NULL means 1, the milestone step.
points	A <code>kits_link_points()</code> result, or a data.frame with <code>id</code> , <code>item</code> , <code>from</code> , <code>to</code> (and <code>se_from</code> , <code>se_to</code> for <code>spread = "se"</code>), for example points from elsewhere.
cut	The cutoff for dropping, in units of spread (a positive number). At least 3 points must be kept.
spread	The spread of the screening: "sd" (default, the SD of the residuals), "mad" (1.4826 x their MAD) or "se" (the SE of each point, from <code>thr_se</code> ; for small samples only).
unit	What is dropped: "step" (default, the flagged points) or "item" (every point of an item with a flagged point).
from_scale, to_scale	In <code>kits_link_screen()</code> : the labels of the two scales of your own points (optional; only with a data.frame). In <code>kits_link_points()</code> : <code>to_scale</code> as in <code>kits_linking()</code> .
screened	A <code>kits_link_screen()</code> result.
link	A link: a <code>kits_link_line()</code> result (or any <code>kits_linking()</code> result, whose linking error is then computed again).
n_boot	Number of item resamples; 0 removes the linking error. 200 is enough to see its size; use more (e.g. 2000) for reporting.
seed	Random seed for the resampling, or NULL for a run that cannot be repeated. Your current random state is left unchanged.

Value

- `kits_link_points()`: a `kits_link_points`, one row per point, `id`, `item`, `from`, `to` (and `step`, `domain`, `se_from`, `se_to` when available).

- `kits_link_screen()`: a `kits_link_screen`, the points with resid, z, flagged and dropped.
- `kits_link_line()`: a `kits_link` (see [kits_linking\(\)](#)) with NA linking errors.
- `kits_link_error()`: the link with `intercept_error`, `slope_error`, `n_failed`, `draws` and `vcov`.

See Also

[kits_linking\(\)](#) (all four steps in one call), [kits_convert\(\)](#).

Examples

```
# Two calibrations with common items a3, a4, a5
betas <- list(a1 = -2, a2 = -1, a3 = c(-0.5, 0.5), a4 = 0, a5 = c(0.5, 1.5), a6 = 2, a7 = 3)
young <- kits_sim(rnorm(500, 0), betas[1:5], seed = 1)
old <- kits_sim(rnorm(500, 2), betas[3:7], prefix = "q", seed = 2)
item_young <- kits_item(young, scale = "young cohort")
item_old <- kits_item(old, scale = "old cohort")

points <- kits_link_points(item_old, item_young) # 1 the points
screened <- kits_link_screen(points)           # 2 one round of screening
link <- kits_link_line(screened)                # 3 the line
link <- kits_link_error(link, n_boot = 200, seed = 1) # 4 the linking error
link

# Your own points: point 8 is shifted and dropped
own <- data.frame(id = paste0("p", 1:8), item = paste0("p", 1:8),
                  from = c(-2, -1, 0, 1, 2, 3, 0.5, 1.5),
                  to = c(-1, 0, 1, 2, 3, 4, 1.5, 5.5))
kits_link_screen(own, from_scale = "x", to_scale = "y", spread = "mad")$dropped
```

<code>kits_linking</code>	<i>Link two scales: between two calibrations, or to D through anchor items</i>
---------------------------	--

Description

Fits a straight mean/sigma line, new value = intercept + slope x old value, with one round of screening, and (with `n_boot > 0`) a linking error that resamples whole items. Between two calibrations each common step is one point; to an external scale each anchor item is one point (its pass step on this scale against its tau). Several domains can share one line. The four steps are also functions of their own ([kits_link_steps](#)). Details: `kits_algorithm("link")` and [tutorial_algorithms/03_linking.html](#).

Usage

```
kits_linking(
  from,
  to,
```

```

step = NULL,
cut = 2,
n_boot = 0,
seed = 1,
to_scale = NULL,
spread = c("sd", "mad", "se"),
unit = c("step", "item")
)

```

Arguments

from	The calibration to be converted: a <code>kits_item()</code> result (for example the M36 cohort), a <code>kits()</code> , its thresholds table (optionally with a domain column), or a named list of them, one per domain.
to	The target: a calibration (<code>kits_item()</code> , <code>kits()</code> or a thresholds table), or a data.frame of anchor items with <code>item_id</code> , <code>tau</code> and optionally <code>step</code> (the pass step).
step	Which step of each item gives the points. Between two calibrations: <code>NULL</code> (every common step) or one whole number. To anchor items: the pass step, "pass" is a score of at least <code>step</code> ; <code>NULL</code> means 1, the milestone step.
cut	The cutoff for dropping, in units of spread (a positive number). At least 3 points must be kept.
n_boot	Number of item resamples for the bootstrap linking error; 0 (default) skips it. 200 is enough to see its size; use more (e.g. 2000) for reporting.
seed	Random seed for the resampling, or <code>NULL</code> for a run that cannot be repeated. Your current random state is left unchanged.
to_scale	The label of the target scale. Between two calibrations it is the scale of <code>to</code> (a different label stops with <code>kits_error_scale</code>); to anchor items it is "D" unless given.
spread	The spread of the screening: "sd" (default, the SD of the residuals), "mad" (1.4826 x their MAD) or "se" (the SE of each point, from <code>thr_se</code> ; for small samples only).
unit	What is dropped: "step" (default, the flagged points) or "item" (every point of an item with a flagged point).

Value

A `kits_link`: a list that prints briefly (`print.kits_link()`). Its elements:

- `intercept`, `slope`: the line after screening (the B and A of Kolen & Brennan, 2014).
- `intercept_error`, `slope_error`, `n_failed`: the linking error (NA when `n_boot` = 0).
- `draws`, `vcov` (only when `n_boot` > 0): the resampled lines and their 2 x 2 covariance.
- `intercept0`, `slope0`: the line before screening.
- `n`: the number of points.
- `dropped`: the dropped points.

- points: one row per point: id, item, from, to, resid, resid_final, z, flagged, dropped (and se_from, se_to, step, domain when they apply).
- settings: cut, n_boot, seed, spread, unit, kind, step (and key).
- from_scale, to_scale: the scales the line converts from and to (NA when not given).
- diagnostics: the checks of the link (see `kits_check()`); info: see `kits_info()`.
- n_items and, between two calibrations, items_total.
- domains (with domains): domain, n, intercept, and the shared slope.

References

Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd ed.). Springer. doi:10.1007/9781493903177

Examples

```
# Each cohort answers part of the items; a3, a4, a5 are common
betas <- list(a1 = -2, a2 = -1, a3 = c(-0.5, 0.5), a4 = 0, a5 = c(0.5, 1.5), a6 = 2, a7 = 3)
young <- kits_sim(rnorm(500, 0), betas[1:5], seed = 1)
old <- kits_sim(rnorm(500, 2), betas[3:7], prefix = "q", seed = 2)
item_old <- kits_item(old, scale = "old cohort")
item_young <- kits_item(young, scale = "young cohort")
link <- kits_linking(item_old, item_young, n_boot = 200)
link

# The older cohort's theta on the younger cohort's scale
s_old <- kits_subj(old, item_old)
head(kits_convert(s_old, link))

# To D through anchor items, two domains sharing one line
set.seed(1)
steps <- data.frame(item_id = paste0("a", 1:12), item_score = 1, thr = rnorm(12),
                    domain = rep(c("lan", "cog"), each = 6))
anchors <- data.frame(item_id = steps$item_id, tau = 64 + 7 * steps$thr + rnorm(12, 0, 0.5))
d_link <- kits_linking(steps, anchors, n_boot = 200)
d_link
kits_convert(c(-1, 0, 1), d_link, from = "my scale")
```

kits_methods

Confidence intervals, covariances and other base methods for kits results

Description

The methods of R's own generics, so the usual names work:

- `confint()`: confidence intervals. For a `kits_item()`, of each threshold: thr +/- z thr_se (normal; the SE is in `coef(item)`). For a link (`kits_linking()`), of the intercept and the slope: with bootstrap draws (`n_boot > 0`) the quantiles of the draws, otherwise none (NA). For a `kits_dif()`, of each item's dif: dif +/- z SE, with the SE from the covariances of the two calibrations (`attr(, "se")`).

- `vcov()`: the covariance matrix of the step parameters beta of a `kits_item()` (rows and columns item:score; 0 for fixed steps).
- `update()`: fit again with some settings changed, for example `update(norm, degrees = c(4, 3, 2, 1))` or `update(item, scale = "lan M03")`. The call recorded in `kits_info()` is run again in the calling environment, so the data must still be there under the same name (results keep no data); data given by value (recorded as `<data>`) cannot be refitted.
- `logLik()`, `nobs()`, and so `AIC()` and `BIC()`: for a `kits_norms()`, the log-likelihood of the shash fit (summed over the groups of a norm by group), its number of coefficients and of person-waves; for a `kits_item()`, the conditional log-likelihood of the CML, the number of free step parameters and of person-waves. Compare only fits of the same data (for example norms of different degrees).
- `simulate()`: responses from the step parameters of a `kits_item()` at given theta, as `kits_sim()` does; `kits_sim()` needs no calibration.

Usage

```
## S3 method for class 'kits_item'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'kits_link'
confint(object, parm = c("intercept", "slope"), level = 0.95, ...)

## S3 method for class 'kits_dif'
confint(object, parm = NULL, level = 0.95, ...)

## S3 method for class 'kits_item'
vcov(object, ...)

## S3 method for class 'kits_item'
logLik(object, ...)

## S3 method for class 'kits_item'
nobs(object, ...)

## S3 method for class 'kits_norms'
logLik(object, ...)

## S3 method for class 'kits_norms'
nobs(object, ...)

## S3 method for class 'kits_item'
simulate(object, nsim = 1, seed = NULL, theta, ...)

## S3 method for class 'kits_norms'
update(object, ...)

## S3 method for class 'kits_item'
update(object, ...)
```

Arguments

object	A kits result (see above).
parm	Which rows: item ids (a <code>kits_item()</code> or a <code>kits_dif()</code>), or "intercept", "slope" (a link); by default all.
level	The confidence level.
...	For <code>update()</code> : the settings to change. Otherwise not used.
nsim	Number of simulated response matrices.
seed	Random seed, as for <code>kits_sim()</code> .
theta	For <code>simulate()</code> : the abilities of the simulated person-waves, on the calibration scale.

Value

`confint()`: a data.frame with the estimate, lower and upper. `vcov()`: a matrix. `update()`: the new result. `logLik()`: a logLik object. `nobs()`: a number. `simulate()`: a response matrix (a list of them when `nsim > 1`).

Examples

```

betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(rnorm(400, 0, 1.5), betas, seed = 1)
item <- kits_item(X, scale = "lan M03")
confint(item)
vcov(item)[1:2, 1:2]
logLik(item)
AIC(item)
X_new <- simulate(item, theta = rnorm(5), seed = 2)
X_new

set.seed(1)
age <- runif(600, 3, 36)
D <- 20 + 15 * log(age) + rnorm(600, 0, 3)
norm <- kits_norms(D, age, ages = 3:36, degrees = c(3, 2, 1, 1))
lower <- update(norm, degrees = c(2, 1, 1, 1))
AIC(norm, lower)
nobs(norm)

```

kits_norm_ages

Age range of the norms

Description

From the 1st percentile of age in the first wave to the 99th percentile of age in the last wave, both ends rounded outward to by (by default half a month). Pass only the waves to be normed (screen them first with `kits_norm_screen()`).

Usage

```
kits_norm_ages(age_m, wave_m, probs = c(0.01, 0.99), by = 0.5)
```

Arguments

age_m	Age in months of each person-wave, or a <code>kits_subj()</code> result (its <code>age_m</code> and <code>wave_m</code> are used, for example <code>kits_norm_ages(s[screen\$normed,])</code>).
wave_m	The wave (months) of each person-wave (with a <code>kits_subj</code> : taken from its column).
probs	Which percentile to take in the first wave and in the last wave.
by	The age interval.

Details

Details: `kits_algorithm("norm_screen")` and `tutorial_algorithms/04_norms.html`.

Value

A vector of ages, which can be passed directly to ages of `kits_norms()`.

Examples

```
set.seed(1)
age <- c(rnorm(200, 12, 0.6), 7, 8, rnorm(200, 24, 0.6))
wave_m <- c(rep(12, 202), rep(24, 200))
range(kits_norm_ages(age, wave_m))

# The same from a kits_subj with age_m and wave_m columns
X <- kits_sim(rnorm(402), list(a1 = c(-1, 0), a2 = 0.5, a3 = -0.5), seed = 1)
persons <- data.frame(visit_id = rownames(X), age_m = age, wave_m = wave_m)
s <- kits_subj(X, kits_item(X), persons = persons)
range(kits_norm_ages(s))
```

kits_norm_screen	<i>Screening waves before fitting norms: waves with scores piled at the floor</i>
------------------	---

Description

Computes, for each wave, the percentage of children at the lowest score of that wave (D rounded to 3 decimals), and leaves a wave out of the norms when it reaches cut. Waves piled at the ceiling still go into the norms.

Usage

```
kits_norm_screen(d, wave_m, cohort = NULL, cut = 20, ...)
```

```
## S3 method for class 'kits_norm_screen'
plot(x, ...)
```

Arguments

d	The score of each person-wave (for example the D score), or a kits_subj() result (its theta, wave_m and cohort are used; a message says when its scale is not "D").
wave_m	The wave (months) of each person-wave (with a kits_subj: taken from its column).
cohort	The cohort of each person-wave; optional, the cohorts are then computed separately (with a kits_subj: taken from its column when present).
cut	A wave whose percentage at the lowest score reaches this number is left out of the norms; the percentage is first rounded to 1 decimal (as printed in floor_pct).
...	Not used: an argument the function does not have stops with an error (class kits_error_input).
x	A kits_norm_screen.

Details

Details: `kits_algorithm("norm_screen")` and `tutorial_algorithms/04_norms.html`.

Value

A `kits_norm_screen`: a list that prints briefly. Its elements:

- waves: one row per (cohort x) wave: cohort (when given), wave_m, n, floor_pct (the percentage at the lowest score), normed (FALSE means left out of the norms).
- normed: one TRUE/FALSE per person-wave, in input order (NA in d is FALSE).
- cut, diagnostics (see [kits_check\(\)](#)) and info (see [kits_info\(\)](#)).

Examples

```
D <- c(10, 10, 10, 10, 12, 15, 20, 22, 25, 30, 31, 33)
wave_m <- rep(c(6, 12), each = 6)
screen <- kits_norm_screen(D, wave_m)
screen
screen$normed

# The two cohorts computed separately
kits_norm_screen(D, wave_m, cohort = rep(c("M03", "M36"), times = 6))

# The share at the lowest score of each wave; the waves left out in orange
p <- plot(screen)
if (interactive()) p
```

kits_norm_steps	<i>A norm step by step: data, fit, table</i>
-----------------	--

Description

The three steps of `kits_norms()`, each a function; with the same settings they give the same table:

1. `kits_norm_data()`: the person-waves with finite score and age within ages plus pad;
2. `kits_norm_fit()`: the shash by maximum likelihood, one fit per start of delta, the best kept;
3. `kits_norm_table()`: the shash parameters and percentiles at each age, with the diagnostics.

Usage

```
kits_norm_data(d, age_m, ages, pad = 1, age_x = sqrt, ...)
```

```
## S3 method for class 'kits_norm_data'
print(x, digits = 2, n = NULL, ...)
```

```
kits_norm_fit(
  norm_data,
  degrees = c(5, 4, 3, 2),
  delta_starts = c(0.5, 1, 1.5, 2)
)
```

```
## S3 method for class 'kits_norm_fit'
print(x, ...)
```

```
kits_norm_table(fit, pct = NORM_PCT, delta_max = 3)
```

Arguments

<code>d</code>	The score of each person-wave (for example the D score). For a <code>kits_subj()</code> result <code>s</code> , give <code>s\$theta</code> and <code>s\$age_m</code> .
<code>age_m</code>	Age in months of each person-wave.
<code>ages</code>	The ages at which the norms will be given.
<code>pad</code>	Use data up to this many months outside the range of ages.
<code>age_x</code>	The transformation function for age.
<code>...</code>	In <code>kits_norm_data()</code> : not used (an argument the function does not have stops with an error, class <code>kits_error_input</code>). In <code>print()</code> : not used.
<code>x</code>	A <code>kits_norm_data</code> or a <code>kits_norm_fit</code> .
<code>digits</code>	Decimals to print (only the printout is rounded).
<code>n</code>	Rows to print (see <code>print.kits_table()</code>).
<code>norm_data</code>	A <code>kits_norm_data()</code> result.

degrees	The degrees of the four polynomials for mu, sigma, epsilon, delta (see kits_norms()); at least $10 * \text{sum}(\text{degrees} + 1)$ person-waves are needed (kits_error_data).
delta_starts	Start values for delta.
fit	A kits_norm_fit() result.
pct	The percentiles to output.
delta_max	When the largest delta exceeds this number, a warning is shown.

Details

Details: [kits_algorithm\("norms"\)](#) and [tutorial_algorithms/04_norms.html](#).

Value

- [kits_norm_data\(\)](#): a [kits_norm_data](#), a data.frame with d and age_m, one row per person-wave used; `attr(, "norm")` keeps ages, ages_in, pad, age_x, age_x_text, scale.
- [kits_norm_fit\(\)](#): a [kits_norm_fit](#), a list with model (the polynomial coefficients, -log L value, convergence), data (the [kits_norm_data\(\)](#)), degrees and delta_starts.
- [kits_norm_table\(\)](#): a [kits_norms](#) (see [kits_norms\(\)](#)).

See Also

[kits_norms\(\)](#) (all three steps in one call), [kits_norm_z\(\)](#).

Examples

```
set.seed(1)
age <- runif(600, 3, 36)
D <- 20 + 15 * log(age) + rnorm(600, 0, 3)

# 1 the data inside the ages
norm_data <- kits_norm_data(D, age, ages = seq(3, 36, by = 3))
norm_data

# 2 the shash by maximum likelihood
fit <- kits_norm_fit(norm_data, degrees = c(3, 2, 1, 1))
fit

# 3 the norm table
norm <- kits_norm_table(fit)
norm
```

kits_norm_z	<i>Same-age z-score</i>
-------------	-------------------------

Description

Looks up the shash parameters for each age in the norm table (interpolating between rows, each segment on its own) and returns the same-age z; ages outside the norm range give NA. A perfect score has no upper limit, so its z is not meaningful: set it to NA yourself (see the examples). With a norm by group, each child is put on the curves of its group.

Usage

```
kits_norm_z(d, ...)  
  
## Default S3 method:  
kits_norm_z(d, age_m, norm, by = NULL, ...)  
  
## S3 method for class 'kits_subj'  
kits_norm_z(d, norm, ...)
```

Arguments

- d Scores (for example the D score), or a `kits_subj()` result (its theta and age_m are used; its scale must match the norm's, else `kits_error_scale`).
- ... Not used: an argument the function does not have stops with an error (class `kits_error_input`).
- age_m Age in months (not with a `kits_subj`).
- norm The result of `kits_norms()`. Several segments can be stacked together, marked by a segment column.
- by With scores as numbers and a norm by group: a data.frame of the group columns of the norm, one row per score (a `kits_subj()` needs these columns itself).

Details

Details: `kits_algorithm("norms")` and `tutorial_algorithms/04_norms.html`.

Value

With scores as numbers: the z-scores (the percentile rank is $100 * \text{pnorm}(z)$). With a `kits_subj()`: the scores with the columns `z` and `pct` ($100 * \text{pnorm}(z)$).

Examples

```
norm <- data.frame(age_m = c(6, 12), mu = c(40, 60), sigma = 5, epsilon = 0, delta = 1)  
kits_norm_z(c(40, 50, 66), age_m = c(6, 9, 12), norm = norm)  
  
# Ages outside the range get no z
```

```

kits_norm_z(50, age_m = 20, norm = norm)

# Perfect scores: no z
raw <- c(3, 5, 8)
max_raw <- c(8, 8, 8)
z <- kits_norm_z(c(40, 50, 66), age_m = c(6, 9, 12), norm = norm)
z[raw == max_raw] <- NA
z

```

kits_norms

Fit a continuous norm (shash by maximum likelihood in RTMB)

Description

Fits a sinh-arcsinh (shash) distribution of the scores whose four parameters are polynomials in transformed age (by default the square root), and returns the norm table: the parameters and the percentiles at each age of ages. Scores can be 0 or negative. Norms are given only within the age range of the data, rounded out to whole months. Screen the waves first with [kits_norm_screen\(\)](#); the three steps are [kits_norm_steps](#).

Usage

```

kits_norms(d, ...)

## Default S3 method:
kits_norms(
  d,
  age_m,
  ages,
  degrees = c(5, 4, 3, 2),
  age_x = sqrt,
  pct = NORM_PCT,
  pad = 1,
  delta_starts = c(0.5, 1, 1.5, 2),
  delta_max = 3,
  by = NULL,
  ...
)

## S3 method for class 'kits_subj'
kits_norms(
  d,
  ages,
  degrees = c(5, 4, 3, 2),
  age_x = sqrt,
  pct = NORM_PCT,
  pad = 1,
  delta_starts = c(0.5, 1, 1.5, 2),

```

```

    delta_max = 3,
    by = NULL,
    ...
)

## S3 method for class 'kits_norms'
print(x, digits = 2, n = NULL, ...)
```

Arguments

d	The score of each person-wave (for example the D score), or a kits_subj() result (its theta and age_m are used; a message says when its scale is not "D").
...	Passed to <code>print()</code> for a data.frame.
age_m	Age in months of each person-wave (not with a kits_subj: taken from its column).
ages	The ages at which to output the norms. With a kits_subj() it can be left out: then the rule of kits_norm_ages() sets them, with pad = 0 unless pad is given.
degrees	The degrees of the four polynomials for mu, sigma, epsilon, delta. The default is for about 12 waves; with 2 to 5 waves use for example <code>c(3, 2, 1, 1)</code> , by sex <code>c(4, 3, 2, 1)</code> . At least $10 * \text{sum}(\text{degrees} + 1)$ person-waves must be inside the ages (<code>kits_error_data</code>).
age_x	The transformation function for age.
pct	The percentiles to output.
pad	Use data up to this many months outside the range of ages (with pad = 0, data outside are not used).
delta_starts	Start values for delta: one fit per value, the largest likelihood is kept.
delta_max	When the kept solution's delta exceeds this number, a warning is shown.
by	Optional groups, one norm each: with a kits_subj() , the names of its group columns (for example "sex", or <code>c("domain", "sex")</code>); with scores as numbers, a data.frame of the group columns (one row per score). No group may be NA.
x	A kits_norms.
digits	Decimals to print (only the printout is rounded; the object keeps full precision).
n	Rows to print: by default all rows of a table of up to 30 rows, else the first 10 (see print.kits_table()).

Details

Details: `kits_algorithm("norms")` and [tutorial_algorithms/04_norms.html](#); the models tutorial ([tutorial_models/kits_models.html](#)), section 8.

Value

A kits_norms (a data.frame; `plot()` draws the percentile curves) with one row per age:

- age_m, and mu, sigma, epsilon, delta: location, scale, skewness, tail weight.

- `n`, `neg_loglik` (the -log-likelihood of the kept solution).
- the percentiles P3, P5, ...
- `attr(, "kits_args")`: the settings (`ages_in`, `degrees`, `pad`, `delta_starts`, `age_x`, `pct`, `delta_max`, as in `kitsPreview`).
- `attr(, "scale")`, `attr(, "diagnostics")` (`kits_check()`), `attr(, "info")` (`kits_info()`).
- With `by`: the group columns first; `attr(, "by")` names them, `attr(, "groups")` keeps each norm.

Pass it to `kits_norm_z()` for same-age z-scores.

Examples

```
set.seed(1)
age <- runif(600, 3, 36)
D <- 20 + 15 * log(age) + rnorm(600, 0, 3)
norm <- kits_norms(D, age, ages = seq(3, 36, by = 3), degrees = c(3, 2, 1, 1))
library(dplyr)
norm |>
  select(age_m, P10, P50, P90)

# The same norm in three steps
norm_data <- kits_norm_data(D, age, ages = seq(3, 36, by = 3))
fit <- kits_norm_fit(norm_data, degrees = c(3, 2, 1, 1))
all.equal(kits_norm_table(fit)$P50, norm$P50)
# The children binned: no single child in the plot
p <- plot(norm, scores = data.frame(theta = D, age_m = age))
if (interactive()) p

# One norm per sex: one table, the group column first
sex <- rep(c("boy", "girl"), 300)
by_sex <- kits_norms(D, age, ages = seq(3, 36, by = 3), degrees = c(3, 2, 1, 1),
  by = data.frame(sex = sex))
by_sex |>
  select(sex, age_m, P50)
kits_norm_z(c(50, 60), age_m = c(12, 24), by_sex, by = data.frame(sex = c("boy", "girl")))
```

kits_pcm_prob

PCM category probabilities, expected scores and information given theta

Description

A teaching view of the partial credit model (Masters, 1982) at given abilities: for each item and θ , the category probabilities $P(X_i = x \mid \theta)$, the expected score $E(X_i \mid \theta)$ and the item information $\text{Var}(X_i \mid \theta)$, from the same core as the rest of kits; their sums over the items equal `kits_expected_score()` and `kits_information()`.

Usage

```

kits_pcm_prob(item, theta, items = NULL)

## S3 method for class 'kits_pcm_prob'
print(x, ...)

## S3 method for class 'kits_pcm_prob'
as.data.frame(x, ...)

## S3 method for class 'kits_pcm_prob'
plot(x, type = c("category", "expected", "information"), items = NULL, ...)

```

Arguments

item	The item parameters: a <code>kits_item()</code> result (or its fit, a <code>kits()</code> , or a <code>data.frame</code> with <code>item_id</code> , <code>item_score</code> , <code>beta</code>).
theta	The abilities (a numeric vector), on the calibration scale of item (logits).
items	Optional item ids (default: all items of item). For <code>plot()</code> : the items to draw (default: all items of x); <code>type = "category"</code> draws the first of them.
x	A <code>kits_pcm_prob</code> object.
...	Not used.
type	"category" (the default): the category curves $P(X = x \mid \theta)$ of one item; "expected": the expected test score (blue) and item scores (gray); "information": the test information (blue), its slope $I'(\theta)$ (orange, dashed) and the item information (gray).

Details

Details: the models tutorial (tutorial_models/kits_models.html), sections 1 and 9.1.

Value

A `kits_pcm_prob` object, a list with

- `prob`: long, one row per item, theta and score: `item_id`, `theta`, `x`, `prob`;
- `moments`: one row per item and theta: `item_id`, `theta`, `expected` ($E(X_i \mid \theta)$), `variance` ($\text{Var}(X_i \mid \theta)$, the item information), `third` (the third central moment);
- `test`: one row per theta: `theta`, `expected` (the expected raw score), `information`, `information_slope` ($I'(\theta)$); also `as.data.frame(x)`;
- `items`, `steps` (the betas used), `scale`, `info`.

References

Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47(2), 149-174.
[doi:10.1007/BF02296272](https://doi.org/10.1007/BF02296272)

See Also

`kits_esf()` ($P(X_i = x | r)$), `kits_information()`, `kits_expected_score()`, `kits_algorithm()`.

Examples

```
# One item with two steps, by hand
parms <- data.frame(item_id = "a1", item_score = 1:2, beta = c(-1, 0.5))
pp <- kits_pcm_prob(parms, theta = c(-2, 0, 2))
pp
pp$prob

# Two items of the language calibration of kitm03
item <- kitm03$responses |>
  dplyr::filter(domain == "lan") |>
  kits_resp() |>
  kits_item(scale = "lan M03")
pp <- kits_pcm_prob(item, seq(-6, 6, by = 0.1), items = c("lana01", "lana02"))
p <- plot(pp, items = "lana02")
if (interactive()) p
```

kits_precision

Precision of each form across the scale

Description

For each form (a set of items), how well does the WLE recover theta at each point of the scale? At each theta of a grid it gives the expected WLE, its bias, SD and RMSE, from the raw-score distribution of the Lord-Wingersky recursion (Thissen et al., 1995; Kolen & Brennan, 2014), and the usable range of each form, where $RMSE < criterion$. Details: `kits_algorithm("information")` and `tutorial_algorithms/06_scale.html`.

Usage

```
kits_precision(
  item,
  forms,
  theta = seq(-6, 6, by = 0.1),
  link = NULL,
  criterion = NULL
)

## S3 method for class 'kits_precision'
print(x, digits = 2, ...)

## S3 method for class 'kits_precision'
plot(x, ...)
```

Arguments

item	The item parameters: a <code>kits_item()</code> result (its step parameters beta, on its calibration scale), or a <code>kits()</code> , or a data.frame with item_id, item_score, beta.
forms	The forms: a <code>kits_subj_table()</code> result (the standard form of each wave, from <code>attr(, "forms")</code>), a named list of item patterns (character vectors of item ids), or one item pattern.
theta	The grid, on the calibration scale of item (logits).
link	Optional: a link or a list of links (applied in order) to report the results on another scale, for example <code>list(co, d_link)</code> for D.
criterion	The largest RMSE of the usable range, on the scale of the result. NULL (the default) means 1 logit of the calibration scale, that is the product of the slopes of link.
x	A kits_precision.
digits	Decimals to print.
...	Not used.

Value

A table (a data.frame) with one row per form and theta: form, theta, expected ($E[\theta - \hat{\theta}]$), bias, sd, rmse; on the scale in `attr(, "scale")`. `attr(, "ranges")`: one row per form, form, n_items, max_raw, from, to (the usable range; NA when no point of the grid reaches the criterion), min_rmse and at (where it is smallest). `attr(, "criterion")`, diagnostics and `kits_info()` too. `plot()` draws the RMSE curves.

References

- Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd ed.). Springer. doi:10.1007/9781493903177
- Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. L. (1995). Item response theory for scores on tests including polytomous items with ordered responses. *Applied Psychological Measurement*, 19(1), 39-49. doi:10.1177/014662169501900105

Examples

```
betas <- lapply(seq(-3, 3, length.out = 12), function(b) b + c(-0.5, 0.5))
names(betas) <- sprintf("a%02d", 1:12)
X <- kits_sim(rnorm(800, 0, 1.5), betas, seed = 1)
item <- kits_item(X)
pr <- kits_precision(item, list(easy = sprintf("a%02d", 1:8), hard = sprintf("a%02d", 5:12)),
                             criterion = 0.6)

pr
p <- plot(pr)
if (interactive()) p
```

kits_resp

*Responses with the information of each child***Description**

`kits_resp()` puts the responses of one domain and the information of each child in one table, one row per person-wave: first the columns of the child (`visit_id`, `subj_id`, `cohort`, `sex`, `wave_m`, `age_m` and any other column that does not change within a person-wave), then one column per item. `kits` remembers which columns are items (`attr(, "items")`). The later functions (`kits_item()`, `kits_subj()`, `kits_item_table()`, `kits_dif()`) take the information of the child from this table; it is optional, and `filter()` and `mutate()` keep it a `kits_resp`. `print()` gives the counts and the checks, `summary()` the standard form of each wave, and `plot()` the design (which items each wave answered). Details: `vignette("kits-flow", package = "kits")`, section 0.

Usage

```
kits_resp(data, persons = NULL)

## S3 method for class 'kits_resp'
print(x, ...)

## S3 method for class 'kits_resp'
summary(object, ...)

## S3 method for class 'kits_resp'
plot(x, ...)
```

Arguments

<code>data</code>	Responses: a long <code>data.frame</code> (<code>visit_id</code> , <code>item_id</code> , <code>item_score</code> and the columns of the child; one domain at a time), or a matrix or a wide <code>data.frame</code> (see <code>kits_item()</code>).
<code>persons</code>	Optional information of each person-wave, a <code>data.frame</code> with <code>visit_id</code> (one row per person-wave; extra rows are fine). Its columns are added to the table.
<code>x, object</code>	A <code>kits_resp</code> .
<code>...</code>	Not used.

Value

A `kits_resp`: a `data.frame` with one row per person-wave, the columns of the child first, then the items; `attr(, "items")` (the item columns), `attr(, "domain")` (when the data had one), `diagnostics` and `kits_info()`.

See Also

`kits_item()`, `kits_subj()`, `kits_item_table()`, `kits_dif()`.

Examples

```
library(dplyr)
resp <- kitm03$responses |>
  filter(domain == "lan", wave_m <= 12) |>
  kits_resp()
resp
summary(resp)
p <- plot(resp)
if (interactive()) p

# The information of the child goes along
item <- kits_item(resp, scale = "lan M03")
scores <- kits_subj(resp, item)
head(scores)

# A matrix with the information given separately
X <- kits_sim(rnorm(200), list(a1 = 0, a2 = c(-1, 1), a3 = 0.5), seed = 1)
persons <- data.frame(visit_id = rownames(X), wave_m = rep(c(6, 12), each = 100))
kits_resp(X, persons = persons)
```

kits_save_plot

Save a plot: interactive html and static png

Description

The html is an interactive plot (zoom, hover); the png is a browser screenshot of the same plot. Saving html needs pandoc (RStudio and Quarto come with it); when it is not found, the function stops (class `kits_error_input`) and says how to set `RSTUDIO_PANDOC`. Saving png needs the `webshot2` package and a Chromium-based browser (its path in `CHROMOTE_CHROME`).

Usage

```
kits_save_plot(p, html, png = NULL, width = 1200, height = 800)
```

Arguments

<code>p</code>	A plotly plot.
<code>html</code>	The html file name; it must end in <code>.html</code> .
<code>png</code>	The png file name. Omit it when no png is wanted.
<code>width, height</code>	Width and height of the png (pixels).

Value

The saved file names (invisibly).

Examples

```
set.seed(1)
s <- data.frame(cohort = rep(c("M03", "M36"), each = 200), wave_m = 36,
                 theta = rnorm(400, 68, 4))
p <- plot(kits_vertical(s), type = "median")

# Save html only (needs pandoc)
if (rmarkdown::pandoc_available()) kits_save_plot(p, file.path(tempdir(), "cohort.html"))

## Not run:
# Also save png (needs webshot2 and a browser)
kits_save_plot(p, file.path(tempdir(), "cohort.html"), file.path(tempdir(), "cohort.png"),
               width = 800, height = 420)

## End(Not run)
```

kits_sim	<i>Simulate PCM responses</i>
----------	-------------------------------

Description

Simulates the scores (starting at 0) of persons with abilities θ on PCM items with step parameters β , for small examples.

Usage

```
kits_sim(theta, betas, prefix = NULL, seed = NULL)
```

Arguments

theta	Ability of each person.
betas	A list with one element per item: the names are item ids, each element holds the step parameters of that item (one beta for a dichotomous item).
prefix	Start of the person ids ("y0001" for prefix = "y"); give each cohort its own. NULL (default): "p", 4 hex digits made from the simulated data, and "_" ("p3fa2_0001").
seed	Random seed; when given, your current random state is left unchanged. NULL (default) uses the current random state, so you can call <code>set.seed()</code> first.

Value

A matrix: one row per person and one column per item; row names are `visit_id`, column names are item ids. It can be passed directly to the functions in this package.

Examples

```
betas <- list(a1 = 0, a2 = c(-1, 1))
kits_sim(theta = c(-2, 0, 2), betas = betas)

# Two cohorts: give each its own prefix, so the ids differ
young <- kits_sim(rnorm(5), betas, prefix = "y", seed = 1)
old <- kits_sim(rnorm(5, 1), betas, prefix = "o", seed = 2)
rownames(rbind(young, old))
```

kits_subj

Person abilities (WLE) that can always be computed

Description

Estimates the weighted likelihood estimate (WLE; Warm, 1989) of every person-wave and its SE, finite also for a zero or a perfect score, plus person fit. `theta` is on the scale of `item` (recorded in `attr(, "scale")`); convert it with `kits_convert()`. The result keeps the raw-score tables of its item patterns, so `scores |> kits_subj_table()` gives the table of each wave. For the PCM the WLE equals the mode of the likelihood penalized by the Jeffreys prior (Firth, 1993; Magis, 2015); `kits` computes it by maximizing $\log L(\theta) + 1/2 \log I(\theta)$ with RTMB (`algo_jme()`). With two maxima (few items with very different difficulties) the higher one is taken (the objective value decides); with two equally high maxima, the smaller `theta`. The SE is $1 / \sqrt{I(\theta)}$, the asymptotic SE of the WLE (Warm, 1989).

Usage

```
kits_subj(data, item, persons = NULL, ...)

## S3 method for class 'kits_subj'
print(x, digits = 2, n = NULL, ...)
```

Arguments

<code>data</code>	Responses, as in <code>kits_item()</code> : a matrix, a wide or long data.frame, or a <code>kits_resp()</code> table (its columns of the child are then the persons). One domain only; gaps in the observed scores are allowed.
<code>item</code>	The item parameters: the result of <code>kits_item()</code> (or its fit), or a <code>kits()</code> , or a data.frame with <code>item_id</code> , <code>item_score</code> , <code>beta</code> .
<code>persons</code>	Optional person information to add to the result: a data.frame with <code>visit_id</code> and any of <code>subj_id</code> (the child id, the same in every wave), <code>wave_m</code> (wave, months), <code>age_m</code> (age, months), <code>sex</code> and <code>cohort</code> ; other columns are dropped. Every person-wave needs a row.
<code>...</code>	In <code>kits_subj()</code> : not used (an unknown argument stops with an error). In <code>print()</code> : passed to <code>print()</code> for a data.frame.
<code>x</code>	A <code>kits_subj</code> .
<code>digits</code>	Decimals to print (only the printout is rounded; the object keeps full precision).
<code>n</code>	Rows to print: by default all rows of a table of up to 30 rows, else the first 10.

Details

Details: `kits_algorithm("wle")` and the tutorial, section wle; the models tutorial (tutorial_models/kits_models.html), section 4.

Value

A data.frame with one row per person, in the order of data:

- `pattern_id`, `visit_id`: the item pattern (the combination of items answered) and the id.
- `raw`: the raw score.
- `raw_max`: the highest possible raw score on the items answered (`raw == raw_max`: perfect).
- `theta`, `se`: the WLE and its SE, $1 / \sqrt{I(\theta)}$.
- `infit`, `outfit`: person fit, the mean squares of this person's residuals (NA for a zero or a perfect score; much below 1 with few items, so compare persons of the same form).
- `flag`: "zero" (zero score), "perfect" (perfect score), "misfit" (infit or outfit above 2), or "ok".
- With persons, then its columns `subj_id`, `wave_m`, `age_m`, `sex`, `cohort` (those present).

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
item <- kits_item(X)
head(kits_subj(X, item))

# Add the person information: the norm functions then take the result directly
library(dplyr)
persons <- tibble(visit_id = rownames(X), wave_m = rep(c(12, 24), each = 150)) |>
  mutate(age_m = wave_m + runif(n(), -0.5, 0.5))
s <- kits_subj(X, item, persons = persons)
head(s)
summary(s)

# Or the responses and the child's columns in one table (kits_resp)
resp <- kitm03$responses |>
  filter(domain == "lan", wave_m <= 6) |>
  kits_resp()
item_resp <- kits_item(resp, scale = "lan M03")
head(kits_subj(resp, item_resp))

# dplyr verbs keep the scale of the scores
s |>
  filter(flag == "ok") |>
  summarise(mean = mean(theta), .by = wave_m)
```

kits_subj_table	<i>Raw-score to ability table</i>
-----------------	-----------------------------------

Description

scores |> kits_subj_table() gives the table of each wave from the scores of `kits_subj()`: the WLE and the count of every raw score of the standard form (the item pattern answered by most children of the wave), on the scale of the scores. `kits_subj_table(item)` gives the WLE of every raw score of every item pattern of a calibration, without data.

Usage

```
kits_subj_table(x, ...)

## S3 method for class 'kits_pattern_table'
print(x, digits = 2, n = NULL, ...)

## S3 method for class 'kits_subj_table'
print(x, digits = 2, n = NULL, ...)
```

Arguments

- x The scores of `kits_subj()` (converted or not, filtered or stacked), or the item parameters: a `kits_item()` result, a `kits()`, its fit (`kits_cml()`), or a data.frame with item_id, item_score, beta. In `print()`: a `kits_subj_table`.
- ... In `kits_subj_table()`: not used (an unknown argument stops with an error). In `print()`: passed to `print()` for a data.frame.
- digits Decimals to print (only the printout is rounded; the object keeps full precision).
- n Rows to print: by default all rows of a table of up to 30 rows, else the first 10.

Details

Details: tutorial_algorithms/02_scoring.html.

Value

From scores: a `kits_subj_table` (a data.frame); one row is one raw score of the standard form in one wave: wave_m, n_items, raw, theta, se (and se_link, se_total when the conversions add them), n (children with this raw score on the standard form), pct, cum_pct, wave_n (all children in the wave), form_pct (percentage of the wave’s children on the standard form). `attr(, "forms")` lists every item pattern of each wave: wave_m, form (1 = the standard form), type ("missing responses" or "other items"), n, items. It prints with a note on the children who are not in the table, and keeps a diagnostics table.

From item parameters: a table (a data.frame) with one row per item pattern and per raw score (0 to the maximum): pattern_id, raw, theta, se.

Examples

```

betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
X[1:10, "a4"] <- NA
item <- kits_item(X)

# The standard form of each wave, from the scores
persons <- data.frame(visit_id = rownames(X), wave_m = rep(c(12, 24), each = 150))
scores <- kits_subj(X, item, persons = persons)
scores |>
  kits_subj_table()

# Every raw score of every item pattern, without data
kits_subj_table(item)

# A table of item parameters: all its items are one item pattern
parms <- data.frame(item_id = c("a1", "a1", "a2"), item_score = c(1, 2, 1), beta = c(-1, 0, 0.5))
kits_subj_table(parms)

```

kits_vertical	<i>Statistics of a vertical scale: growth, variability and separation by wave</i>
---------------	---

Description

For scores of every cohort on one scale (for example D, after [kits_convert\(\)](#)), gives per cohort x wave the location and spread (with `sd_true`, the SD without measurement error), the growth from the previous wave, and its separation (effect_size of Yen, 1986; `p_sup`), the properties of Kolen and Brennan (2014). `attr(, "cohorts")` compares cohorts at shared waves. `plot()` draws the mean +/- SD (or the median and quartiles) at the median age of each wave. Details: [tutorial_algorithms/06_scale.html](#).

Usage

```

kits_vertical(s, extremes = TRUE)

## S3 method for class 'kits_vertical'
print(x, digits = 2, ...)

## S3 method for class 'kits_vertical'
plot(
  x,
  type = c("mean", "median"),
  cohorts = NULL,
  age_ticks = AGE_TICKS,
  y_range = NULL,
  ...
)

```

Arguments

s	Scores on one scale: a <code>kits_subj()</code> result (converted) with <code>wave_m</code> and optionally <code>cohort</code> , <code>age_m</code> , <code>se</code> , <code>se_link</code> and <code>flag</code> ; or a <code>data.frame</code> with <code>theta</code> and <code>wave_m</code> and any of those.
extremes	Use the children with a zero or a perfect score (<code>flag</code>) in <code>sd_true</code> ? TRUE (the default) uses every child; FALSE leaves them out of both the SD and the mean error variance.
x	A <code>kits_vertical</code> .
digits	Decimals to print.
...	Not used.
type	What <code>plot()</code> draws: "mean" (the mean +/- SD) or "median" (the median, bars from P25 to P75).
cohorts	Which cohorts to draw, in this order (the first blue circles, the second orange squares); by default all of them.
age_ticks	Age ticks.
y_range	The range of the y axis, the same in every panel; by default from the data.

Value

A table (a `data.frame`) with one row per cohort x wave, in the order of cohort and wave: `cohort` (when given), `wave_m`, `n`, `age_med` (median age, when `age_m` is given), `mean`, `sd`, `median`, `P25`, `P75`, `sd_true` (NA without `se`, or with fewer than 2 children without a zero or perfect score), `growth`, `effect_size`, `p_sup` (NA for the first wave of a cohort), and `se_link` (the linking error of the mean, when the scores have `se_link`). `attr("cohorts")` compares cohorts at shared waves; `diagnostics` and `kits_info()` too.

References

- Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking* (3rd ed.). Springer. doi:10.1007/9781493903177
- Yen, W. M. (1986). The choice of scale for educational measurement: An IRT perspective. *Journal of Educational Measurement*, 23(4), 299-325. doi:10.1111/j.17453984.1986.tb00252.x

Examples

```
library(dplyr)
set.seed(1)
s <- tibble(cohort = rep(c("M03", "M36"), c(600, 400)),
            wave_m = c(rep(c(12, 24, 36), each = 200), rep(c(36, 48), each = 200))) |>
  mutate(theta = 40 + 8 * log(wave_m) + rnorm(n(), 0, 4), se = 2)
v <- kits_vertical(s)
v
attr(v, "cohorts")
p <- plot(v, type = "median")
if (interactive()) p

# Two domains stacked: one panel per domain
```

```
both <- list(lan = v, motor = kits_vertical(mutate(s, theta = theta + 5))) |>
  purrr::list_rbind(names_to = "domain")
p <- plot(both)
```

NORM_PCT

Default percentiles of the norms

Description

The default pct of `kits_norms()` and `kits_norm_table()`.

Usage

```
NORM_PCT
```

Format

A numeric vector: 3, 5, 10, 15, 25, 50, 75, 85, 90, 95, 97.

Examples

```
NORM_PCT
```

plot.kits_norms

Plot a continuous norm: percentile curves, one norm or by group

Description

One norm: the percentile curves, with the children's scores optionally drawn behind them (binned by default, so the plot and its html file hold no single child). A norm by group (`kits_norms(..., by = )`): one panel per domain x group, each group in its own colour.

Usage

```
## S3 method for class 'kits_norms'
plot(
  x,
  scores = NULL,
  pct = c(3, 10, 50, 90, 97),
  persons = c("bins", "points"),
  min_cell = 5,
  age_ticks = AGE_TICKS,
  y_range = NULL,
  ...
)
```

Arguments

x	The result of <code>kits_norms()</code> (it can have a segment column).
scores	Optional: the scores of the children (a <code>kits_subj()</code> result, or a data.frame with theta and age_m; for a norm by group also the group columns), drawn behind the curves.
pct	Which percentiles to draw; they must be columns of x.
persons	How to draw the children: "bins" (default, counts per cell of a 40 x 40 grid) or "points" (one point per child; for your own checks only, not for plots you share).
min_cell	With persons = "bins", cells with fewer children than this are not drawn (default 5).
age_ticks	By group: the age ticks.
y_range	By group: the range of the y axis, the same in every panel; by default one common range from the data of all panels (rounded to multiples of 10).
...	Not used.

Value

A plotly plot.

See Also

`plot()` of a `kits_item_table()`: the growth curves with the items.

Examples

```
norm <- data.frame(age_m = c(6, 12, 18), mu = c(40, 55, 62), sigma = 5, epsilon = 0, delta = 1,
                  P3 = c(31, 46, 53), P50 = c(40, 55, 62), P97 = c(49, 64, 71))
class(norm) <- c("kits_norms", "kits_table", "data.frame")
p <- plot(norm, pct = c(3, 50, 97))
if (interactive()) p

# One norm per sex: one panel per sex
set.seed(1)
age <- runif(1200, 3, 24)
sex <- rep(c("boy", "girl"), 600)
D <- 20 + 15 * log(age) + ifelse(sex == "girl", 1, 0) + rnorm(1200, 0, 3)
by_sex <- kits_norms(D, age, ages = 3:24, degrees = c(3, 2, 1, 1), by = data.frame(sex = sex))
p <- plot(by_sex)
if (interactive()) p
```

print.kits_link	<i>Print or plot a link</i>
-----------------	-----------------------------

Description

print() shows the line, the linking error (when n_boot > 0), and the dropped points; coef() gives the intercept and the slope. plot() draws every point (from against to) with the final line, the line before screening and the screening band (dashed); dropped points are marked.

Usage

```
## S3 method for class 'kits_link'
print(x, digits = 2, ...)

## S3 method for class 'kits_link'
coef(object, ...)

## S3 method for class 'kits_link'
plot(x, ...)
```

Arguments

x	The result of <code>kits_linking()</code> or <code>kits_link_line()</code> .
digits	Decimals to print (only the printout is rounded; the object keeps full precision).
...	Not used.
object	A link.

Value

print(): x (invisibly). plot(): a plotly plot. coef(): c(intercept = , slope =). \$, [[: the element.

Examples

```
points <- data.frame(id = paste0("p", 1:6), item = paste0("p", 1:6),
                     from = c(-2, -1, 0, 1, 2, 3), to = c(-1, 0, 1, 2, 3.2, 6))
link <- kits_link_line(kits_link_screen(points))
link
coef(link)
p <- plot(link)
if (interactive()) p
```

print.kits_norm_screen	<i>Print a norm screen</i>
------------------------	----------------------------

Description

Print a norm screen

Usage

```
## S3 method for class 'kits_norm_screen'  
print(x, digits = 2, ...)
```

Arguments

x	The result of kits_norm_screen() .
digits	Decimals to print (only the printout is rounded; the object keeps full precision).
...	Not used.

Value

x (invisibly).

Examples

```
kits_norm_screen(c(10, 10, 10, 10, 12, 15, 20, 22, 25, 30, 31, 33), rep(c(6, 12), each = 6))
```

print.kits_table	<i>Print a kits table</i>
------------------	---------------------------

Description

Tables from kits (for example [kits_subj\(\)](#), [kits_itemfit\(\)](#), [kits_subj_table\(\)](#), [coef\(\)](#) of [kits_item\(\)](#)) are data.frames that print digits decimals (2 by default; the object keeps its full precision); a table that keeps diagnostics ends with one line, "Checks: ...". A table of up to 30 rows prints in full, a longer one its first 10 rows (`print(x, n = 50)` prints more, as `data.frame(x)` gives all rows). Selecting rows or columns and dplyr verbs keep the class and the recorded scale, and drop the diagnostics and the provenance. `rbind()` and `dplyr::bind_rows()` check that the recorded scales match: different scales stop with an error of class `kits_error_scale`, a table without a recorded scale gives a warning of class `kits_warning_scale`; `summary()` and [kits_check\(\)](#) work on the stacked table. Details: `vignette("kits-flow", package = "kits")`, section 4.

Usage

```
## S3 method for class 'kits_table'
print(x, digits = 2, n = NULL, ...)

## S3 method for class 'kits_itemfit'
print(x, digits = 2, n = NULL, ...)

## S3 method for class 'kits_table'
x[i, j, ...]

## S3 method for class 'kits_table'
rbind(..., deparse.level = 1)

## S3 method for class 'kits_subj'
rbind(..., deparse.level = 1)
```

Arguments

x	A table from kits.
digits	Decimals to print.
n	Rows to print. By default all rows of a table of up to 30 rows, and the first 10 rows of a longer table. Inf prints every row.
...	Passed to print() for a data.frame, for example row.names = FALSE.
i, j	Rows and columns, as for a data.frame.
deparse.level	Passed to rbind().

Value

print(): x (invisibly). rbind(): the stacked table.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
item <- kits_item(X)
coef(item)
print(coef(item), digits = 4)
coef(item)$thr[1]

# A long table prints its first rows
s <- kits_subj(X, item)
s
print(s, n = 3)

# rbind() keeps the scale when the scales match
nrow(rbind(s[1:5, ], s[6:10, ]))

# dplyr verbs keep the class and the scale
```

```
library(dplyr)
s |>
  filter(flag == "ok") |>
  attr("scale")
```

summary.kits

Summary of a kits result: a short print and the diagnostics

Description

Prints a result with its whole diagnostics table. For a link, theta adds how much the screening moves the converted values (the line before and after screening at theta); for scores converted by [kits_convert\(\)](#), the summary also shows the links applied, the line they make together and, when every link has draws, the linking error at the mean theta.

Usage

```
## S3 method for class 'kits_item'
summary(object, digits = 2, ...)

## S3 method for class 'kits_table'
summary(object, digits = 2, ...)

## S3 method for class 'kits'
summary(object, digits = 2, ...)

## S3 method for class 'kits_link'
summary(object, digits = 2, theta = NULL, ...)

## S3 method for class 'kits_drift'
summary(object, digits = 2, ...)

## S3 method for class 'kits_norm_screen'
summary(object, digits = 2, ...)

## S3 method for class 'kits_summary'
print(x, ...)

## S3 method for class 'kits_link_check'
summary(object, digits = 2, ...)
```

Arguments

object	A result of kits: kits_item() , kits_subj() , kits_itemfit() , kits_subj_table() , kits() , a link (kits_linking() , kits_link_line()), kits_drift() , kits_norm_screen() , kits_norms() , kits_dif() , kits_vertical() , kits_precision() or kits_link_check() .
digits	Decimals to print.

...	Not used.
theta	For a link: optional values on the scale the link starts from, for example the median theta of each wave.
x	A kits_summary.

Details

Details: [tutorial_algorithms/05_checks.html](https://tutorial.algorithms/05_checks.html).

Value

A kits_summary (printed): object and diagnostics, and

- for a link with theta: screening, a table with theta, before (intercept0 + slope0 theta), after (intercept + slope theta), shift, and with draws se_link and $z = \text{shift} / \text{se_link}$, on the scale where the link ends (attr(, "scale"));
- for converted scores: conversion, a list with chain (one row per link: step, from, to, intercept, slope), the composed intercept and slope, from_scale, to_scale, theta_mean, value_mean and se_link (NA when a link has no draws); one list per calibration for stacked scores.

Examples

```
betas <- list(a1 = c(-1, 0), a2 = c(0, 1), a3 = 0.5, a4 = -0.5)
X <- kits_sim(seq(-3, 3, length.out = 300), betas, seed = 1)
summary(kits_item(X))
summary(kits_subj(X, kits_item(X)))

# Converted scores: the links applied, as one line, and the linking error at the mean
item <- kits_item(X, scale = "x")
anchors <- data.frame(item_id = c("a1", "a2", "a3", "a4"), tau = c(42, 50, 54, 46))
d_link <- kits_linking(item, anchors, n_boot = 50)
s <- kits_subj(X, item) |>
  kits_convert(d_link)
summary(s)
summary(s)$conversion$se_link

# A link: how much the screening moves the converted values
points <- data.frame(id = paste0("p", 1:8), item = paste0("p", 1:8),
  from = c(-2, -1, 0, 1, 2, 3, 0.5, 1.5),
  to = c(-1, 0, 1, 2, 3, 4, 1.5, 5.5))
link <- points |>
  kits_link_screen() |>
  kits_link_line() |>
  kits_link_error(n_boot = 100)
summary(link, theta = c(-1, 0, 1, 2))
summary(link, theta = c(-1, 0, 1, 2))$screening
```

Index

- * **algo**
 - algo_cml, 4
 - algo_esf, 5
 - algo_jme, 6
 - algo_line, 7
 - algo_link_error, 8
 - algo_lw, 9
 - algo_mean_sigma, 10
 - algo_pcm, 11
 - algo_screen, 12
 - algo_shash, 13
 - algo_shash_z, 14
 - algo_thresholds, 15
- * **datasets**
 - kitm03, 17
 - [.kits_table (print.kits_table), 76
- AGE_TICKS, 4
- algo_cml, 4
- algo_cml(), 5, 7–15
- algo_esf, 5
- algo_esf(), 5, 7–15
- algo_jme, 6
- algo_jme(), 5, 7–15, 68
- algo_line, 7
- algo_line(), 5, 7–15
- algo_link_error, 8
- algo_link_error(), 5, 7, 9–15
- algo_lw, 9
- algo_lw(), 5, 7, 8, 10–15
- algo_mean_sigma, 10
- algo_mean_sigma(), 5, 7–9, 11–15
- algo_pcm, 11
- algo_pcm(), 5, 7–10, 12–15
- algo_screen, 12
- algo_screen(), 5, 7–11, 13–15
- algo_shash, 13
- algo_shash(), 5, 7–12, 14, 15
- algo_shash_z, 14
- algo_shash_z(), 5, 7–13, 15
- algo_thresholds, 15
- algo_thresholds(), 5, 7–14
- as.data.frame.kits, 15
- as.data.frame.kits_drift
 - (as.data.frame.kits), 15
- as.data.frame.kits_esf (kits_esf), 30
- as.data.frame.kits_info
 - (as.data.frame.kits), 15
- as.data.frame.kits_item
 - (as.data.frame.kits), 15
- as.data.frame.kits_link
 - (as.data.frame.kits), 15
- as.data.frame.kits_link_check
 - (as.data.frame.kits), 15
- as.data.frame.kits_norm_fit
 - (as.data.frame.kits), 15
- as.data.frame.kits_norm_screen
 - (as.data.frame.kits), 15
- as.data.frame.kits_pcm_prob
 - (kits_pcm_prob), 61
- as.data.frame.kits_stats
 - (as.data.frame.kits), 15
- base::as.data.frame(), 16
- coef.kits_item (kits_item), 35
- coef.kits_link (print.kits_link), 75
- confint.kits_dif (kits_methods), 51
- confint.kits_item (kits_methods), 51
- confint.kits_link (kits_methods), 51
- kitm03, 17
- kitm36 (kitm03), 17
- kits, 19
- kits(), 17, 20, 31, 34, 38, 44, 48, 50, 62, 64, 68, 70, 78
- kits_algorithm, 20
- kits_algorithm(), 31, 40, 63
- kits_anchors, 21
- kits_anchors(), 18

kits_check, 22
 kits_check(), 15, 27, 30, 37, 47, 51, 55, 61, 76
 kits_cml(kits_item_steps), 39
 kits_cml(), 4, 31, 34, 36, 70
 kits_convert, 24
 kits_convert(), 7, 20, 32, 42, 49, 68, 71, 78
 kits_dif, 25
 kits_dif(), 19, 20, 23, 28, 36, 42, 51, 53, 65, 78
 kits_dif_combine, 28
 kits_dif_combine(), 27, 42
 kits_drift, 29
 kits_drift(), 16, 20, 23, 27, 78
 kits_esf, 30
 kits_esf(), 5, 63
 kits_expected_score(kits_information), 33
 kits_expected_score(), 61, 63
 kits_info, 32
 kits_info(), 17, 23, 27, 30, 37, 40, 47, 51, 52, 55, 61, 64, 65, 72
 kits_information, 33
 kits_information(), 20, 61, 63
 kits_item, 35
 kits_item(), 4, 15, 17, 19, 20, 22–24, 26, 29–32, 34, 35, 38–40, 42, 44, 46–48, 50–53, 62, 64, 65, 68, 70, 76, 78
 kits_item_restscore, 38
 kits_item_restscore(), 30, 31
 kits_item_steps, 39
 kits_item_table, 41
 kits_item_table(), 4, 23, 28, 36, 65, 74
 kits_itemfit, 44
 kits_itemfit(), 22, 27, 30, 31, 36, 37, 39, 76, 78
 kits_link_check, 45
 kits_link_check(), 17, 23, 78
 kits_link_error(kits_link_steps), 47
 kits_link_error(), 8
 kits_link_line(kits_link_steps), 47
 kits_link_line(), 10, 16, 75, 78
 kits_link_points(kits_link_steps), 47
 kits_link_points(), 48
 kits_link_screen(kits_link_steps), 47
 kits_link_screen(), 12
 kits_link_steps, 23, 24, 47, 49
 kits_linking, 49
 kits_linking(), 16, 20–24, 29, 32, 46–49, 51, 75, 78
 kits_methods, 51
 kits_norm_ages, 53
 kits_norm_ages(), 60
 kits_norm_data(kits_norm_steps), 56
 kits_norm_fit(kits_norm_steps), 56
 kits_norm_fit(), 13, 17
 kits_norm_screen, 54
 kits_norm_screen(), 17, 20, 23, 53, 59, 76, 78
 kits_norm_steps, 56, 59
 kits_norm_table(kits_norm_steps), 56
 kits_norm_table(), 73
 kits_norm_z, 58
 kits_norm_z(), 14, 57, 61
 kits_norms, 59
 kits_norms(), 4, 20, 23, 32, 42, 52, 54, 56–58, 73, 74, 78
 kits_pcm_prob, 61
 kits_pcm_prob(), 11
 kits_precision, 63
 kits_precision(), 9, 20, 23, 78
 kits_resp, 65
 kits_resp(), 19, 26, 36, 38, 42, 44, 68
 kits_save_plot, 66
 kits_sim, 67
 kits_sim(), 52, 53
 kits_stats(kits_item_steps), 39
 kits_stats(), 4, 17, 36
 kits_subj, 68
 kits_subj(), 6, 19, 20, 22, 24, 32, 36, 39, 54–56, 58, 60, 65, 70, 72, 74, 76, 78
 kits_subj_table, 70
 kits_subj_table(), 6, 22, 24, 64, 76, 78
 kits_thresholds(kits_item_steps), 39
 kits_thresholds(), 15
 kits_vertical, 71
 kits_vertical(), 4, 23, 78
 logLik.kits_item(kits_methods), 51
 logLik.kits_norms(kits_methods), 51
 nobs.kits_item(kits_methods), 51
 nobs.kits_norms(kits_methods), 51
 NORM_PCT, 73
 plot.kits(kits), 19
 plot.kits_dif(kits_dif), 25

plot.kits_drift (kits_drift), 29
 plot.kits_esf (kits_esf), 30
 plot.kits_item (kits_item), 35
 plot.kits_item_restscore
 (kits_item_restscore), 38
 plot.kits_item_table (kits_item_table),
 41
 plot.kits_itemfit (kits_itemfit), 44
 plot.kits_link (print.kits_link), 75
 plot.kits_norm_screen
 (kits_norm_screen), 54
 plot.kits_norms, 73
 plot.kits_pcm_prob (kits_pcm_prob), 61
 plot.kits_precision (kits_precision), 63
 plot.kits_resp (kits_resp), 65
 plot.kits_vertical (kits_vertical), 71
 print.kits (kits), 19
 print.kits_dif (kits_dif), 25
 print.kits_dif_summary (kits_dif), 25
 print.kits_drift (kits_drift), 29
 print.kits_esf (kits_esf), 30
 print.kits_expected_score
 (kits_information), 33
 print.kits_info (kits_info), 32
 print.kits_information
 (kits_information), 33
 print.kits_item (kits_item), 35
 print.kits_item_restscore
 (kits_item_restscore), 38
 print.kits_itemfit (print.kits_table),
 76
 print.kits_link, 75
 print.kits_link(), 50
 print.kits_link_check
 (kits_link_check), 45
 print.kits_norm_data (kits_norm_steps),
 56
 print.kits_norm_fit (kits_norm_steps),
 56
 print.kits_norm_screen, 76
 print.kits_norms (kits_norms), 59
 print.kits_pattern_table
 (kits_subj_table), 70
 print.kits_pcm_prob (kits_pcm_prob), 61
 print.kits_precision (kits_precision),
 63
 print.kits_resp (kits_resp), 65
 print.kits_stats (kits_item_steps), 39
 print.kits_subj (kits_subj), 68
 print.kits_subj_table
 (kits_subj_table), 70
 print.kits_summary (summary.kits), 78
 print.kits_table, 76
 print.kits_table(), 26, 56, 60
 print.kits_vertical (kits_vertical), 71
 rbind.kits_subj (print.kits_table), 76
 rbind.kits_table (print.kits_table), 76
 simulate.kits_item (kits_methods), 51
 summary.kits, 78
 summary.kits_dif (kits_dif), 25
 summary.kits_drift (summary.kits), 78
 summary.kits_item (summary.kits), 78
 summary.kits_link (summary.kits), 78
 summary.kits_link_check (summary.kits),
 78
 summary.kits_norm_screen
 (summary.kits), 78
 summary.kits_resp (kits_resp), 65
 summary.kits_table (summary.kits), 78
 update.kits_item (kits_methods), 51
 update.kits_norms (kits_methods), 51
 vcov.kits_item (kits_methods), 51