Package: measr (via r-universe)

October 29, 2026

Title Bayesian Psychometric Measurement Using 'Stan'

Version 2.0.1

Description Estimate diagnostic classification models (also called cognitive diagnostic models) with 'Stan'. Diagnostic classification models are confirmatory latent class models, as described by Rupp et al. (2010, ISBN: 978-1-60623-527-0). Automatically generate 'Stan' code for the general loglinear cognitive diagnostic diagnostic model proposed by Henson et al. (2009) <doi:10.1007/s11336-008-9089-5> and other subtypes that introduce additional model constraints. Using the generated 'Stan' code, estimate the model evaluate the model's performance using model fit indices, information criteria, and reliability metrics.

License GPL (>= 3)

URL https://measr.r-dcm.org, https://github.com/r-dcm/measr

BugReports https://github.com/r-dcm/measr/issues

Depends R (>= 4.1.0)

Imports bridgesampling, cli, dcm2, dcmstan (>= 0.1.0), dplyr (>= 1.1.1), dtplyr, fs, glue, lifecycle, loo, methods, posterior, psych, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rdcmchecks (>= 0.1.1), rlang (>= 1.1.0), rstan (>= 2.26.0), rstantools (>= 2.6.0), S7, stats, tibble, tidyr (>= 1.3.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

Suggests cmdstanr (>= 0.4.0), dcmdata, knitr, rmarkdown, roxygen2, spelling, testthat (>= 3.0.0), withr

Additional_repositories https://stan-dev.r-universe.dev

Config/testthat/edition 3

Config/Needs/website r-dcm/rdcmtemplate, wjakethompson/wjake, showtext, ggdist, english

Config/Needs/documentation openpharma/roxylint

Config/roxylint list(linters = roxylint::tidy)

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Biarch true

SystemRequirements GNU make

VignetteBuilder knitr

NeedsCompilation yes

Author W. Jake Thompson [aut, cre] (ORCID: <https://orcid.org/0000-0001-7339-0300>), Jeffrey Hoover [aut] (ORCID: <https://orcid.org/0000-0002-0276-0308>), Auburn Jimenez [ctb] (ORCID: <https://orcid.org/0000-0002-7072-2960>), Nathan Jones [ctb] (ORCID: <https://orcid.org/0000-0001-6177-7161>), Matthew Johnson [cph] (Authored code adapted for measrdcm method for `reliability()`), University of Kansas [cph], Institute of Education Sciences [fnd]

Maintainer W. Jake Thompson <wjakethompson@gmail.com>

Config/pak/sysreqs make

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

Date/Publication 2026-03-27 20:59:35 UTC

RemoteUrl https://github.com/cran/measr

RemoteRef HEAD

RemoteSha 5a43293f4331f85331ec9d423df7ef99a731a80f

Contents

aic . . . 3
bayes_factor . . . 4
cdi . . . 6
dcm_estimate . . . 8
fit_ppmc . . . 9
log_mll . . . 12
loglik_array . . . 13
loo-waic . . . 14
m2 . . . 15
measr_examples . . . 16
measr_extract . . . 16
measrdcm . . . 19
model_evaluation . . . 21
qmatrix_validation . . . 24
reliability . . . 26

score 28
stan-classes 29
yens_q3 31

Index **33**

`aic` *Maximum likelihood based information criteria*

Description

Calculate information criteria for diagnostic models not estimated with full Markov chain Monte Carlo (i.e., with `method = "optim"`). Available information include the Akaike information criterion (AIC; Akaike, 1973) and the Bayesian information criterion (BIC; Schwarz, 1978).

Usage

```
aic(x, ..., force = FALSE)

bic(x, ..., force = FALSE)
```

Arguments

`x`	A measrdcm object estimated with `backend = "optim"`.
`...`	Unused.
`force`	If the criterion has already been added to the model object with `add_criterion()`, should it be recalculated. Default is `FALSE`.

Value

The numeric value of the information criterion.

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov & F. Csáki (Eds.), *Proceedings of the Second International Symposium on Information Theory* (pp. 267-281). Akademiai Kiado.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*(2), 461–464. doi:10.1214/aos/1176344136

Examples

```
model_spec <- dcm_specify(
  qmatrix = dcmdata::mdm_qmatrix,
  identifier = "item"
)
model <- dcm_estimate(
  dcm_spec = model_spec,
```

```
  data = dcmdata::mdm_data,
  identifier = "respondent",
  method = "optim",
  seed = 63277
)

aic(model)

bic(model)
```

`bayes_factor` *Bayes factor for model comparisons*

Description

Calculate the Bayes factor for model comparisons, which represents the posterior odds of the null hypothesis when the prior probability of the null model is 0.5 (Jeffreys, 1935; Kass & Raftery, 1995). Consistent with the Bayesian reporting guidelines from Kruschke (2021), we calculate the posterior probability of the null model for a variety of prior probabilities, in addition to the Bayes factor.

Usage

```
bayes_factor(
  x,
  ...,
  model_names = NULL,
  prior_prob = seq(0.02, 0.98, by = 0.02)
)
```

Arguments

`x`	A measrdcm object.
`...`	Additional measrdcm to be compared to `x`.
`model_names`	Names given to each provided model in the comparison output. If `NULL` (the default), the names will be parsed from the names of the objects passed for comparison.
`prior_prob`	A numeric vector of prior probabilities for the null model used to calculate the posterior probability of the null model relative to alternative model. See details for more information.

Details

Bayes factors will be calculated for all possible pairwise comparisons between the models provided to `x` and `....` In each comparison, one model is identified as the null model, and the other is the

alternative. This distinction is not terribly meaningful from a calculation standpoint, as the probabilities for the alternative model are simply 1 minus the null probabilities. If you want particular models to be labeled as the "null", the determination is made by the order the models are sent to the function. That is, `x` will always be the null model. The first model included in `...` will be the alternative model when compared to `x` and the null model when compared to all other models included in `....` Similarly, the second model included in `...` will be the alternative model when compared to `x` and the first model included in `...` and the null model in all other comparisons.

`prior_prob` is used to specify a vector of possible prior probabilities for the null model. These are used in conjunction with the Bayes factor to determine the posterior model probability for the null model, relative to the alternative model. The posterior probability for the alternative model can be calculated as 1 minus the null model's posterior probability. You may specify a specific prior probability, or specify a range of possibilities to construct a graph similar to Kruschke's (2021) Figure 1. These probabilities can be interpreted as, "If the prior probability is {`prior_prob_null`}, then the posterior is {`posterior_prob_null`}" (or 1 minus for the alternative model).

Value

A tibble with one row per model comparison and four columns.

- `null_model`: The null model in the comparison.
- `alt_model`: The alternative model in the comparison.
- `bf`: The estimated Bayes factor.
- `posterior_probs`: A nested list column, where element element is a tibble with two columns:
  - `prior_prob_null`: The prior probability that the null model is correct.
  - `posterior_prob_null`: The posterior probability that the null model is correct.

  The list column can be unnested with `tidyr::unnest()` (see examples). If `prior_prob` is `NULL`, the `posterior_probs` column is excluded from the returned object.

References

Jeffreys, H. (1935). Some tests of significance, treated by the theory of probability. *Mathematical Proceedings of the Cambridge Philosophical Society, 31*(2), 203-222. doi:10.1017/S030500410001330X

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association, 90*(430), 773-795. doi:10.1080/01621459.1995.10476572

Kruschke, J. K. (2021). Bayesian analysis reporting guidelines. *Nature, 5*, 1282-1291. doi:10.1038/s41562021011777

Examples

```
mdm_dina <- dcm_estimate(
  dcm_specify(
    qmatrix = dcmdata::mdm_qmatrix,
    identifier = "item",
    measurement_model = dina()
  ),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
```

```
  method = "mcmc",
  seed = 63277,
  backend = "rstan",
  iter = 700,
  warmup = 500,
  chains = 2,
  refresh = 0
)

mdm_dino <- dcm_estimate(
  dcm_specify(
    qmatrix = dcmdata::mdm_qmatrix,
    identifier = "item",
    measurement_model = dino()
  ),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "mcmc",
  seed = 63277,
  backend = "rstan",
  iter = 700,
  warmup = 500,
  chains = 2,
  refresh = 0
)

bf <- bayes_factor(mdm_dina, mdm_dino)
bf

tidyr::unnest(bf, "posterior_probs")
```

---

cdi *Item, attribute, and test-level discrimination indices*

Description

The cognitive diagnostic index (CDI) is a measure of how well an assessment is able to distinguish between attribute profiles. The index was originally proposed by Henson & Douglas (2005) for item- and test-level discrimination, and then expanded by Henson et al. (2008) to include attribute-level discrimination indices.

Usage

```
cdi(model, weight_prevalence = TRUE)
```

Arguments

`model`	The estimated model to be evaluated.
`weight_prevalence`	Logical indicating whether the discrimination indices should be weighted by the prevalence of the attribute profiles. See details for additional information.

Details

Henson et al. (2008) described two attribute-level discrimination indices, $\mathbf{d}_{(A)\cdot}$ (Equation 8) and $\mathbf{d}_{(B)\cdot}$ (Equation 13), which are similar in that both are the sum of item-level discrimination indices. In both cases, item-level discrimination indices are calculated as the average of Kullback-Leibler information for all pairs of attributes profiles for the item. The item-level indices are then summed to achieve the test-level discrimination index for each attribute, or the test overall. However, whereas $\mathbf{d}_{(A)\cdot}$ is an unweighted average of the Kullback-Leibler information, $\mathbf{d}_{(B)\cdot}$ is a weighted average, where the weight is defined by the prevalence of each profile (i.e., `measr_extract(model, what = "strc_param")`).

Value

A list with two elements:

- `item_discrimination`: A tibble with one row per item containing the CDI for the item and any relevant attributes.
- `test_discrimination`: A tibble with one row containing the total CDI for the assessment and for each attribute.

References

Henson, R., & Douglas, J. (2005). Test construction for cognitive diagnosis. *Applied Psychological Measurement, 29*(4), 262-277. doi:10.1177/0146621604272623

Henson, R., Roussos, L., Douglas, J., & Xuming, H. (2008). Cognitive diagnostic attribute-level discrimination indices. *Applied Psychological Measurement, 32*(4), 275-288. doi:10.1177/0146621607302478

Examples

```
rstn_ecpe_lcdm <- dcm_estimate(
  dcm_specify(dcmdata::ecpe_qmatrix, identifier = "item_id"),
  data = dcmdata::ecpe_data,
  missing = NA,
  identifier = "resp_id",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

cdi(rstn_ecpe_lcdm)
```

dcm_estimate *Fit Bayesian diagnostic classification models*

Description

Estimate diagnostic classification models (DCMs; also known as cognitive diagnostic models) using 'Stan'. Models can be estimated using Stan's optimizer, or full Markov chain Monte Carlo (MCMC).

Usage

```
dcm_estimate(
  dcm_spec,
  data,
  missing = NA,
  identifier = NULL,
  method = c("variational", "mcmc", "optim", "pathfinder"),
  backend = getOption("measr.backend", "rstan"),
  file = NULL,
  file_refit = getOption("measr.file_refit", "never"),
  ...
)
```

Arguments

dcm_spec	A DCM specification created with dcm_specify().
data	Response data. A data frame with 1 row per respondent and 1 column per item.
missing	An R expression specifying how missing data in data is coded (e.g., NA, ".", -99, etc.). The default is NA.
identifier	Optional. Variable name of a column in data that contains respondent identifiers. NULL (the default) indicates that no identifiers are present in the data, and row numbers will be used as identifiers.
method	Estimation method. Options are "variational", which uses Stan's variational algorithm; "mcmc", which uses Stan's sampling method; "optim", which uses Stan's optimizer; or "pathfinder" which uses Stan's pathfinder variational inference algorithm (only available if backend = "cmdstanr").
backend	Character string naming the package to use as the backend for fitting the Stan model. Options are "rstan" (the default) or "cmdstanr". Can be set globally for the current R session via the "measr.backend" option (see options()). Details on the rstan and cmdstanr packages are available at https://mc-stan.org/rstan/ and https://mc-stan.org/cmdstanr/, respectively.
file	Either NULL (the default) or a character string. If a character string, the fitted model object is saved as an .rds object using saveRDS() using the supplied character string. The .rds extension is automatically added. If the specified file already exists, measr will load the previously saved model. Unless file_refit is specified, the model will not be refit.

`file_refit` Controls when a saved model is refit. Options are `"never"`, `"always"`, and `"on_change"`. Can be set globally for the current R session via the "measr.file_refit" option (see `options()`).

- For `"never"` (the default), the fitted model is always loaded if the `file` exists, and model fitting is skipped.
- For `"always"`, the model is always refitted, regardless of whether or not `file` exists.
- For `"on_change"`, the model will be refit if the `dcm_spec`, `data`, `method`, or `backend` specified are different from that in the saved `file`.

`...` Additional arguments passed to Stan.

- For `backend = "rstan"`, arguments are passed to `rstan::sampling()` or `rstan::optimizing()`.
- For `backend = "cmdstanr"`, arguments are passed to the `$sample()` or `$optimize()` methods of the CmdStanModel class.

Value

A `measrdcm` object.

Examples

```
model_spec <- dcm_specify(
  qmatrix = dcmdata::mdm_qmatrix,
  identifier = "item"
)
model <- dcm_estimate(
  dcm_spec = model_spec,
  data = dcmdata::mdm_data,
  identifier = "respondent",
  method = "optim",
  seed = 63277
)
```

---

fit_ppmc *Posterior predictive model checks for assessing model fit*

Description

For models estimated with a method that results in posterior distributions (e.g., "mcmc", "variational"), use the posterior distributions to compute expected distributions for fit statistics and compare to values in the observed data.

Usage

```
fit_ppmc(
  x,
  ...,
  model_fit = NULL,
  item_fit = NULL,
  ndraws = NULL,
  probs = c(0.025, 0.975),
  return_draws = 0,
  force = FALSE
)
```

Arguments

x	An estimated model object (e.g., from dcm_estimate()).
...	Unused. For future extensions.
model_fit	The posterior predictive model checks to compute for an evaluation of model-level fit. If NULL, no model-level checks are computed. See details.
item_fit	The posterior predictive model checks to compute for an evaluation of item-level fit. If NULL, no item-level checks are computed. See details.
ndraws	The number of posterior draws to base the checks on. Must be less than or equal to the total number of posterior draws retained in the estimated model. If NULL (the default) the total number from the estimated model is used.
probs	The percentiles to be computed by the stats::quantile() function for summarizing the posterior distribution for each fit statistic.
return_draws	Number of posterior draws for each specified fit statistic to be returned. This does not affect the calculation of the posterior predictive checks, but can be useful for visualizing the fit statistics. Must be less than ndraws (or the total number of draws if ndraws = NULL). If 0 (the default), only summaries of the posterior are returned (no individual samples).
force	If all requested PPMCs have already been added to the model object using add_fit(), should they be recalculated. Default is FALSE.

Details

Posterior predictive model checks (PPMCs) use the posterior distribution of an estimated model to compute different statistics. This creates an expected distribution of the given statistic, *if our estimated parameters are correct*. We then compute the statistic in our observed data and compare the observed value to the expected distribution. Observed values that fall outside of the expected distributions indicate incompatibility between the estimated model and the observed data.

For DCMs, we currently support PPMCs at the model and item level. At the model level, we calculate the expected raw score distribution (model_fit = "raw_score") as described by Thompson (2019) and Park et al. (2015). At the item level, we can calculate the conditional probability that a respondent in each class provides a correct response (item_fit = "conditional_prob") as described by Thompson (2019) and Sinharay & Almond (2007) or the overall proportion correct for an item (item_fit = "pvalue"), as described by Thompson (2019). We can also calculate the odds

ratio for each pair of items (`item_fit = "odds_ratio"`) as described by Park et al. (2015) and Sinharay et al. (2006).

Value

A list with two elements, "model_fit" and "item_fit". If either `model_fit = NULL` or `item_fit = NULL` in the function call, this will be a one-element list, with the null criteria excluded. Each list element, is itself a list with one element for each specified PPMC containing a tibble. For example if `item_fit = c("conditional_prob", "odds_ratio")`, the "item_fit" element will be a list of length two, where each element is a tibble containing the results of the PPMC. All tibbles follow the same general structure:

- `obs_{ppmc}`: The value of the relevant statistic in the observed data.
- `ppmc_mean`: The mean of the `ndraws` posterior samples calculated for the given statistic.
- Quantile columns: 1 column for each value of `probs`, providing the corresponding quantiles of the `ndraws` posterior samples calculated for the given statistic.
- `samples`: A list column, where each element contains a vector of length `return_draws`, representing samples from the posterior distribution of the calculated statistic. This column is excluded if `return_draws = 0`.
- `ppp`: The posterior predictive p-value. This is the proportion of posterior samples for calculated statistic that are greater than the observed value. Values very close to 0 or 1 indicate incompatibility between the fitted model and the observed data.

References

Park, J. Y., Johnson, M. S., Lee, Y-S. (2015). Posterior predictive model checks for cognitive diagnostic models. *International Journal of Quantitative Research in Education, 2*(3-4), 244-264. doi:10.1504/IJQRE.2015.071738

Sinharay, S., & Almond, R. G. (2007). Assessing fit of cognitive diagnostic models. *Educational and Psychological Measurement, 67*(2), 239-257. doi:10.1177/0013164406292025

Sinharay, S., Johnson, M. S., & Stern, H. S. (2006). Posterior predictive assessment of item response theory models. *Applied Psychological Measurement, 30*(4), 298-321. doi:10.1177/0146621605285517

Thompson, W. J. (2019). *Bayesian psychometrics for diagnostic assessments: A proof of concept* (Research Report No. 19-01). University of Kansas; Accessible Teaching, Learning, and Assessment Systems. doi:10.35542/osf.io/jzqs8

Examples

```
mdm_dina <- dcm_estimate(
  dcm_specify(
    dcmdata::mdm_qmatrix,
    identifier = "item",
    measurement_model = dina()
  ),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "mcmc",
```

```
  seed = 63277,
  backend = "rstan",
  iter = 700,
  warmup = 500,
  chains = 2,
  refresh = 0
)

fit_ppmc(mdm_dina, model_fit = "raw_score")
```

`log_mll` *Log marginal likelihood calculation*

Description

Calculate the log marginal likelihood with bridge sampling (Meng & Wong, 1996). This is a wrapper around `bridgesampling::bridge_sampler()`. Therefore, log marginal likelihood calculation is currently only available for models estimated with `{rstan}` using MCMC.

Usage

```
log_mll(x, ..., force = FALSE)
```

Arguments

`x`	A measrdcm object estimated with `backend = "optim"`.
`...`	Unused.
`force`	If the criterion has already been added to the model object with `add_criterion()`, should it be recalculated. Default is `FALSE`.

Value

The estimate of the log marginal likelihood.

References

Meng, X.-L., & Wong, W. H. (1996). Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistical Sinica, 6*(4), 831-860. https://www.jstor.org/stable/24306045

Examples

```
model_spec <- dcm_specify(
  qmatrix = dcmdata::mdm_qmatrix,
  identifier = "item"
)
model <- dcm_estimate(
```

```
  dcm_spec = model_spec,
  data = dcmdata::mdm_data,
  identifier = "respondent",
  method = "variational",
  seed = 63277
)

log_mll(model)
```

---

loglik_array *Extract the log-likelihood of an estimated model*

Description

The `loglik_array()` methods for measrdcm objects calculates the log-likelihood for an estimated model via the generated quantities functionality in *Stan* and returns the draws of the `log_lik` parameter.

Usage

```
loglik_array(model, ...)
```

Arguments

`model`	A measrdcm object.
`...`	Unused. For future extensions.

Value

A "`draws_array`" object containing the log-likelihood estimates for the model.

Examples

```
rstn_mdm_lcdm <- dcm_estimate(
  dcm_specify(dcmdata::mdm_qmatrix, identifier = "item"),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

loglik_array(rstn_mdm_lcdm)
```

loo-waic *Relative fit for Bayesian models*

Description

For models estimated with MCMC, relative model fit comparisons can be made using the LOO-CV or WAIC indicates (Vehtari et al., 2017). These functions are wrappers for the loo package. See the loo package vignettes for details on the implementation.

Usage

```
## S3 method for class '`measr::measrdcm`'
loo(x, ..., r_eff = NA, force = FALSE)

## S3 method for class '`measr::measrdcm`'
waic(x, ..., force = FALSE)

## S3 method for class '`measr::measrdcm`'
loo_compare(x, ..., criterion = c("loo", "waic"), model_names = NULL)
```

Arguments

x	A measrdcm object.
...	For loo() and waic(), additional arguments passed to loo::loo.array() or loo::waic.array(), respectively. For loo_compare(), additional measrdcm objects to be compared to x.
r_eff	Vector of relative effective sample size estimates for the likelihood (exp(log_lik)) of each observation. This is related to the relative efficiency of estimating the normalizing term in self-normalized importance sampling when using posterior draws obtained with MCMC. If MCMC draws are used and r_eff is not provided then the reported PSIS effective sample sizes and Monte Carlo error estimates can be over-optimistic. If the posterior draws are (near) independent then r_eff=1 can be used. r_eff has to be a scalar (same value is used for all observations) or a vector with length equal to the number of observations. The default value is 1. See the relative_eff() helper functions for help computing r_eff.
force	If the LOO criterion has already been added to the model object with add_criterion(), should it be recalculated. Default is FALSE.
criterion	The name of the criterion to be extracted from the x for comparison.
model_names	Names given to each provided model in the comparison output. If NULL (the default), the names will be parsed from the names of the objects passed for comparison.

Value

For `loo()` and `waic()`, the information criteria returned by `loo::loo.array()` or `loo::waic.array()`, respectively.

For `loo_compare()`, the criterion comparison returned by `loo::loo_compare()`.

References

Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing, 27*(5), 1413-1432. doi:10.1007/s1122201696964

`m2` *Estimate the M_2 fit statistic for diagnostic classification models*

Description

For diagnostic classification models, the M_2 statistic is calculated as described by Hansen et al. (2016) and Liu et al. (2016).

Usage

```
## S3 method for class '`measr::measrdcm`'
fit_m2(model, ..., ci = 0.9, force = FALSE)
```

Arguments

`model`	An estimated diagnostic classification model.
`...`	Unused, for extensibility.
`ci`	The confidence interval for the RMSEA.
`force`	If the M_2 has already been saved to the model object with `add_fit()`, should it be recalculated. Default is `FALSE`.

Value

A data frame created by `dcm2::fit_m2()`.

References

Hansen, M., Cai, L., Monroe, S., & Li, Z. (2016). Limited-information goodness-of-fit testing of diagnostic classification item response models. *British Journal of Mathematical and Statistical Psychology, 69*(3), 225-252. doi:10.1111/bmsp.12074

Liu, Y., Tian, W., & Xin, T. (2016). An application of M_2 statistic to evaluate the fit of cognitive diagnostic models. *Journal of Educational and Behavioral Statistics, 41*(1), 3-26. doi:10.3102/1076998615621293

Examples

```
rstn_mdm_lcdm <- dcm_estimate(
  dcm_specify(dcmdata::mdm_qmatrix, identifier = "item"),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

fit_m2(rstn_mdm_lcdm)
```

---

measr_examples *Determine if code is executed interactively or in pkgdown*

Description

Used for determining examples that shouldn't be run on CRAN, but can be run for the pkgdown website.

Usage

```
measr_examples()
```

Value

A logical value indicating whether or not the examples should be run.

Examples

```
measr_examples()
```

---

measr_extract *Extract components of a* measrfit *object*

Description

Extract model metadata, parameter estimates, and model evaluation results.

Usage

```
measr_extract(model, what, ...)
```

Arguments

`model`	The estimated to extract information from.
`what`	Character string. The information to be extracted. See details for available options.
`...`	Additional arguments passed to each extract method. • `ppmc_interval`: For `what = "odds_ratio_flags"` and `what = "conditional_prob_flags"`, the compatibility interval used for determining model fit flags to return. For example, a `ppmc_interval` of 0.95 (the default) will return any PPMCs where the posterior predictive *p*-value (ppp) is less than 0.025 or greater than 0.975. • `agreement`: For `what = "classification_reliability"`, additional measures of agreement to include. By default, the classification accuracy and consistency metrics defined Johnson & Sinharay (2018) are returned. Additional metrics that can be specified to `agreement` are Goodman & Kruskal's lambda (`lambda`), Cohen's kappa (`kappa`), Youden's statistic (`youden`), the tetrachoric correlation (`tetra`), true positive rate (`tp`), and the true negative rate (`tn`). For `what = "probability_reliability"`, additional measures of agreement to include. By default, the informational reliability index defined by Johnson & Sinharay (2020) is returned. Additional metrics that can be specified to `agreement` are the point biserial reliability index (`bs`), parallel forms reliability index (`pf`), and the tetrachoric reliability index (`tb`), which was originally defined by Templin & Bradshaw (2013).

Details

For diagnostic classification models, we can extract the following information:

Model metadata:

- `prior`: The priors used when estimating the model.
- `classes`: The possible classes or profile patterns. This will show the class label (i.e., the pattern of proficiency) and the attributes included in each class.

Estimated model components:

Model parameters:

- `item_param`: The estimated item parameters. This shows the name of the parameter, the class of the parameter, and the estimated value.
- `strc_param`: The estimated structural parameters. This is the base rate of membership in each class. This shows the class pattern, the attributes present in each class, and the estimated proportion of respondents in each class.
- `attribute_base_rate`: The estimated base rate of attribute proficiency. Calculated from the structural parameters of the classes where each attribute is present.
- `pi_matrix`: The model estimated probability that a respondent in the given class provides a correct response to the item. The output shows the the item (rows), class (columns), and estimated *p*-values.

- `exp_pvalues`: Model expected *p*-values for each item. This is equivalent to the `pi_matrix`, but also includes an "overall" field, which represents the expected *p*-value for each item (i.e., an average of the class-specific *p*-values, weighted by the prevalence of each class).

Respondent results:

- `class_prob`: The probability that each respondent belongs to class (i.e., has the given pattern of proficiency).
- `attribute_prob`: The proficiency probability for each respondent and attribute.

Model fit:

Absolute model fit:

- `m2`: The M_2 fit statistic. See `fit_m2()` for details.
- `rmsea`: The root mean square error of approximation (RMSEA) fit statistic and associated confidence interval. See `fit_m2()` for details.
- `srmsr`: The standardized root mean square residual (SRMSR) fit statistic. See `fit_m2()` for details.
- `ppmc_raw_score`: The observed and posterior predicted chi-square statistic for the raw score distribution. See `fit_ppmc()` for details.
- `ppmc_conditional_prob`: The observed and posterior predicted conditional probabilities of each class providing a correct response to each item. See `fit_ppmc()` for details.
- `ppmc_conditional_prob_flags`: A subset of the PPMC conditional probabilities where the *ppp* is outside the specified `ppmc_interval`.
- `ppmc_odds_ratio`: The observed and posterior predicted odds ratios of each item pair. See `fit_ppmc()` for details.
- `ppmc_odds_ratio_flags`: A subset of the PPMC odds ratios where the *ppp* is outside the specified `ppmc_interval`.
- `ppmc_pvalue`: The observed and posterior predicted proportion of correct responses to each item. See `fit_ppmc()` for details.
- `ppmc_pvalue_flags`: A subset of the PPMC proportion correct values where the *ppp* is outside the specified `ppmc_interval`.

Relative model fit:

- `loo`: The leave-one-out cross validation results. See `loo::loo()` for details.
- `waic`: The widely applicable information criterion results. See `loo::waic()` for details.
- `aic`: The Akaike information criterion results. See `aic()` for details.
- `bic`: The Bayesian information criterion results. See `bic()` for details.

Reliability:

- `pattern_reliability`: The accuracy and consistency of the overall attribute profile classification, as described by Cui et al. (2012).
- `classification_reliability`: The classification accuracy and consistency for each attribute, using the metrics described by Johnson & Sinharay (2018).
- `probability_reliability`: Reliability estimates for the probability of proficiency on each attribute, as described by Johnson & Sinharay (2020).

Value

The extracted information. The specific structure will vary depending on what is being extracted, but usually the returned object is a tibble with the requested information.

References

Cui, Y., Gierl, M. J., & Chang, H.-H. (2012). Estimating classification consistency and accuracy for cognitive diagnostic assessment. *Journal of Educational Measurement, 49*(1), 19-38. doi:10.1111/j.17453984.2011.00158.x

Johnson, M. S., & Sinharay, S. (2018). Measures of agreement to assess attribute-level classification accuracy and consistency for cognitive diagnostic assessments. *Journal of Educational Measurement, 55*(4), 635-664. doi:10.1111/jedm.12196

Johnson, M. S., & Sinharay, S. (2020). The reliability of the posterior probability of skill attainment in diagnostic classification models. *Journal of Educational and Behavioral Statistics, 45*(1), 5-31. doi:10.3102/1076998619864550

Templin, J., & Bradshaw, L. (2013). Measuring the reliability of diagnostic classification model examinee estimates. *Journal of Classification, 30*(2), 251-275. doi:10.1007/s0035701391294

Examples

```
rstn_mdm_lcdm <- dcm_estimate(
  dcm_specify(dcmdata::mdm_qmatrix, identifier = "item"),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

measr_extract(rstn_mdm_lcdm, "strc_param")
```

`measrdcm` *S7 class for measrdcm objects*

Description

The `measrdcm` constructor is exported to facilitate the conversion of other model objects (e.g., `stanfit`) to `measrdcm` objects. We do not expect or recommend calling this function directly, unless you are creating a method for converting to `measrdcm`. Rather, to create a `measrdcm` object, one should use `dcm_estimate()`.

Usage

```
measrdcm(
  model_spec = NULL,
  data = list(),
  stancode = character(0),
  method = stanmethod(),
  algorithm = character(0),
  backend = stanbackend(),
```

```
  model = list(),
  respondent_estimates = list(),
  fit = list(),
  criteria = list(),
  reliability = list(),
  file = character(0),
  version = list()
)
```

Arguments

model_spec	The model specification used to estimate the model.
data	The data used to estimate the model.
stancode	The model code in *Stan* language.
method	The method used to fit the model.
algorithm	The name of the algorithm used to fit the model.
backend	The name of the backend used to fit the model.
model	The fitted Stan model. This will object of class rstan::stanfit if backend = "rstan" and CmdStanMCMC if backend = "cmdstanr" was specified when fitting the model.
respondent_estimates	An empty list for adding estimated person parameters after fitting the model.
fit	An empty list for adding model fit information after fitting the model.
criteria	An empty list for adding information criteria after fitting the model.
reliability	An empty list for adding reliability information after fitting the model.
file	Optional name of a file which the model objects was saved to or loaded from.
version	The versions of measr, *Stan*, and rstan or cmdstanr that were used to fit the model.

Value

A measrdcm object.

See Also

dcm_estimate().

Examples

```
qmatrix <- tibble::tibble(
  att1 = sample(0:1, size = 15, replace = TRUE),
  att2 = sample(0:1, size = 15, replace = TRUE),
  att3 = sample(0:1, size = 15, replace = TRUE),
  att4 = sample(0:1, size = 15, replace = TRUE)
)

spec <- dcm_specify(qmatrix = qmatrix)

measrdcm(spec)
```

model_evaluation *Add model evaluation metrics model objects*

Description

Add model evaluation metrics to fitted model objects. These functions are wrappers around other functions that compute the metrics. The benefit of using these wrappers is that the model evaluation metrics are saved as part of the model object so that time-intensive calculations do not need to be repeated. See Details for specifics.

Usage

```
add_criterion(
  x,
  criterion = c("loo", "waic", "log_mll", "aic", "bic"),
  overwrite = FALSE,
  save = TRUE,
  ...,
  r_eff = NA
)

add_reliability(x, overwrite = FALSE, save = TRUE, ...)

add_fit(
  x,
  method = c("m2", "ppmc"),
  ndraws = NULL,
  return_draws = ndraws,
  overwrite = FALSE,
  save = TRUE,
  ...,
  ci = 0.9
)

add_respondent_estimates(
  x,
  probs = c(0.025, 0.975),
  overwrite = FALSE,
  save = TRUE
)
```

Arguments

x	A measrdcm object.
criterion	A vector of information criteria to calculate and add to the model object. Must be "loo", "waic", or "log_mll" for models estimated with MCMC, or "aic" or "bic" for models estimated with the optimizer.

`overwrite` Logical. Indicates whether specified elements that have already been added to the estimated model should be overwritten. Default is `FALSE`.

`save` Logical. Only relevant if a file was specified in the measrdcm object passed to `x`. If `TRUE` (the default), the model is re-saved to the specified file when new criteria are added to the `R` object. If `FALSE`, the new criteria will be added to the `R` object, but the saved file will not be updated.

`...` Arguments passed on to `fit_ppmc`

`model_fit` The posterior predictive model checks to compute for an evaluation of model-level fit. If `NULL`, no model-level checks are computed. See details.

`item_fit` The posterior predictive model checks to compute for an evaluation of item-level fit. If `NULL`, no item-level checks are computed. See details.

`r_eff` Vector of relative effective sample size estimates for the likelihood (`exp(log_lik)`) of each observation. This is related to the relative efficiency of estimating the normalizing term in self-normalized importance sampling when using posterior draws obtained with MCMC. If MCMC draws are used and `r_eff` is not provided then the reported PSIS effective sample sizes and Monte Carlo error estimates can be over-optimistic. If the posterior draws are (near) independent then `r_eff=1` can be used. `r_eff` has to be a scalar (same value is used for all observations) or a vector with length equal to the number of observations. The default value is 1. See the `relative_eff()` helper functions for help computing `r_eff`.

`method` A vector of model fit methods to evaluate and add to the model object.

`ndraws` The number of posterior draws to base the checks on. Must be less than or equal to the total number of posterior draws retained in the estimated model. If `NULL` (the default) the total number from the estimated model is used.

`return_draws` Number of posterior draws for each specified fit statistic to be returned. This does not affect the calculation of the posterior predictive checks, but can be useful for visualizing the fit statistics. Must be less than `ndraws` (or the total number of draws if `ndraws = NULL`). If `0` (the default), only summaries of the posterior are returned (no individual samples).

`ci` The confidence interval for the RMSEA, computed from the M2

`probs` The percentiles to be computed by the `stats::quantile()` function. Only relevant if the model was estimated with a method that results in posterior distributions (e.g., "mcmc", "variational"). Only used if `summary` is `TRUE`.

Details

For `add_respondent_estimates()`, estimated person parameters are added to the `@respondent_estimates` element of the fitted model.

For `add_fit()`, model and item fit information are added to the `@fit` element of the fitted model. This function wraps `fit_m2()` to calculate the M_2 statistic (Hansen et al., 2016; Liu et al., 2016) and/or `fit_ppmc()` to calculate posterior predictive model checks (Park et al., 2015; Sinharay & Almond, 2007; Sinharay et al., 2006; Thompson, 2019), depending on which methods are specified.

For `add_criterion()`, relative fit criteria are added to the `@criteria` element of the fitted model. For models estimated with MCMC, this function wraps `loo()` or `waic()` to calculate the LOO-CV

(Vehtari et al., 2017) or WAIC (Watanabe, 2010), respectively, or `log_mll()` to calculate the log marginal likelihood, which is used for calculating Bayes factors. For models estimated with the optimizer, this wraps `aic()` or `bic()` to estimate the AIC (Akaike, 1973) or BIC (Schwarz, 1978), respectively.

For `add_reliability()`, reliability information is added to the `@reliability` element of the fitted model. Pattern level reliability is described by Cui et al. (2012). Classification reliability and posterior probability reliability are described by Johnson & Sinharay (2018, 2020), respectively. This function wraps `reliability()`. Arguments supplied to `...` are passed to `reliability()`.

Value

A modified measrdcm object with the corresponding slot populated with the specified information.

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov & F. Csáki (Eds.), *Proceedings of the Second International Symposium on Information Theory* (pp. 267-281). Akademiai Kiado.

Cui, Y., Gierl, M. J., & Chang, H.-H. (2012). Estimating classification consistency and accuracy for cognitive diagnostic assessment. *Journal of Educational Measurement, 49*(1), 19-38. doi:10.1111/j.17453984.2011.00158.x

Hansen, M., Cai, L., Monroe, S., & Li, Z. (2016). Limited-information goodness-of-fit testing of diagnostic classification item response models. *British Journal of Mathematical and Statistical Psychology, 69*(3), 225-252. doi:10.1111/bmsp.12074

Johnson, M. S., & Sinharay, S. (2018). Measures of agreement to assess attribute-level classification accuracy and consistency for cognitive diagnostic assessments. *Journal of Educational Measurement, 55*(4), 635-664. doi:10.1111/jedm.12196

Johnson, M. S., & Sinharay, S. (2020). The reliability of the posterior probability of skill attainment in diagnostic classification models. *Journal of Educational and Behavioral Statistics, 45*(1), 5-31. doi:10.3102/1076998619864550

Liu, Y., Tian, W., & Xin, T. (2016). An application of M_2 statistic to evaluate the fit of cognitive diagnostic models. *Journal of Educational and Behavioral Statistics, 41*(1), 3-26. doi:10.3102/1076998615621293

Park, J. Y., Johnson, M. S., Lee, Y-S. (2015). Posterior predictive model checks for cognitive diagnostic models. *International Journal of Quantitative Research in Education, 2*(3-4), 244-264. doi:10.1504/IJQRE.2015.071738

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics, 6*(2), 461–464. doi:10.1214/aos/1176344136

Sinharay, S., & Almond, R. G. (2007). Assessing fit of cognitive diagnostic models. *Educational and Psychological Measurement, 67*(2), 239-257. doi:10.1177/0013164406292025

Sinharay, S., Johnson, M. S., & Stern, H. S. (2006). Posterior predictive assessment of item response theory models. *Applied Psychological Measurement, 30*(4), 298-321. doi:10.1177/0146621605285517

Thompson, W. J. (2019). *Bayesian psychometrics for diagnostic assessments: A proof of concept* (Research Report No. 19-01). University of Kansas; Accessible Teaching, Learning, and Assessment Systems. doi:10.35542/osf.io/jzqs8

Vehtari, A., Gelman, A., & Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing, 27*(5), 1413-1432. doi:10.1007/s1122201696964

Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research, 11*(116), 3571-3594. https://jmlr.org/papers/v11/watanabe10a.html

Examples

```
cmds_mdm_dina <- dcm_estimate(
  dcm_specify(
    dcmdata::mdm_qmatrix,
    identifier = "item",
    measurement_model = dina(),
    priors = c(
      prior(beta(5, 17), type = "slip"),
      prior(beta(5, 17), type = "guess")
    )
  ),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

cmds_mdm_dina <- add_reliability(cmds_mdm_dina)
cmds_mdm_dina <- add_fit(cmds_mdm_dina, method = "m2")
cmds_mdm_dina <- add_respondent_estimates(cmds_mdm_dina)
```

---

qmatrix_validation *Q-matrix validation*

Description

Calculate Q-matrix validation metrics for a fitted model objects using methods described by de la Torre and Chiu (2016). See details for additional information.

Usage

```
qmatrix_validation(x, ..., pvaf_threshold = 0.95)
```

Arguments

x	A measrdcm object.
...	Unused.

`pvaf_threshold` The threshold for proportion of variance accounted for to flag items for appropriate empirical specifications. The default is .95 as implemented by de la Torre and Chiu (2016).

Details

Q-matrix validation is conducted by evaluating the proportion of variance accounted for by different Q-matrix specifications. Following the method described by de la Torre and Chiu (2016), we use the following steps for each item:

1. Calculate the total variance explained if an item measured all possible attributes.
2. For each possible Q-matrix entry, calculate the variance explained if the item measured the given attributes. Calculate the proportion of variance explained (PVAF) as the variance explained by the current Q-matrix entry divided by the variance explained by the saturated entry (Step 1).
3. After computing the PVAF for all possible Q-matrix entries, filter to only those with a PVAF greater than the specified `pvaf_threshold` threshold.
4. Filter the remaining Q-matrix entries to those that measure the fewest number of attributes (i.e., we prefer a more parsimonious model).
5. If there is more than one Q-matrix entry remaining, select the entry with the highest PVAF.

Value

A tibble containing the Q-matrix validation results. There is one row per item with 5 columns:

- The item identifier, as specified in the Q-matrix and used to estimate the model.
- `original_specification`: The original Q-matrix entry for the item.
- `original_pvaf`: The proportion of variance accounted for by the original specification, compared to a specification where the item measures all attributes.
- `empirical_specification`: The Q-matrix specification that measures the fewest attributes with a proportion of variance accounted for over the the specified `pvaf_threshold` threshold. If the original specification is optimal, `empirical_specification` will be `NA`.
- `empirical_pvaf`: The proportion of variance accounted for by the empirical specification, compared to a specification where the item measures all attributes. If the original specification is optimal, `emprirical_pvaf` will be `NA`.

References

de la Torre, J., & Chiu, C.-Y. (2016). A general method of empirical Q-matrix validation. *Psychometrika, 81*(2), 253-273. doi:10.1007/s1133601594678

Examples

```
mod_spec <- dcm_specify(
  qmatrix = dcmdata::ecpe_qmatrix,
  identifier = "item_id",
  measurement_model = dcmstan::lcdm(),
  structural_model = dcmstan::hdcm(
```

```
    hierarchy = "lexical -> cohesive -> morphosyntactic"
  )
)
rstn_ecpe <- dcm_estimate(
  mod_spec,
  data = dcmdata::ecpe_data,
  identifier = "resp_id",
  backend = "rstan",
  method = "optim"
)

q_matrix_validation <- qmatrix_validation(rstn_ecpe)
```

`reliability` *Estimate the reliability of a diagnostic classification model*

Description

For diagnostic classification models, reliability can be estimated at the pattern or attribute level. Pattern-level reliability represents the classification consistency and accuracy of placing students into an overall mastery profile. Rather than an overall profile, attributes can also be scored individually. In this case, classification consistency and accuracy should be evaluated for each individual attribute, rather than the overall profile. This is referred to as the *maximum a posteriori* (MAP) reliability. Finally, it may be desirable to report results as the probability of proficiency or mastery on each attribute instead of a proficient/not proficient classification. In this case, the reliability of the posterior probability should be reported. This is the *expected a posteriori* (EAP) reliability.

Usage

```
reliability(x, ..., threshold = 0.5, force = FALSE)
```

Arguments

`x`	The estimated model to be evaluated.
`...`	Unused. For future extensions.
`threshold`	For `map_reliability`, the threshold applied to the attribute-level probabilities for determining the binary attribute classifications. Should be a numeric vector of length 1 (the same threshold is applied to all attributes), or length equal to the number of attributes. If a named vector is supplied, names should match the attribute names in the Q-matrix used to estimate the model. If unnamed, thresholds should be in the order the attributes were defined in the Q-matrix.
`force`	If reliability information has already been added to the model object with `add_reliability()`, should it be recalculated. Default is `FALSE`.

Details

The pattern-level reliability (`pattern_reliability`) statistics are described in Cui et al. (2012). Attribute-level classification reliability statistics (`map_reliability`) are described in Johnson & Sinharay (2018). Reliability statistics for the posterior mean of the skill indicators (i.e., the mastery or proficiency probabilities; `eap_reliability`) are described in Johnson & Sinharay (2019).

Value

For class `measrdcm`, a list with 3 elements:

- `pattern_reliability`: The pattern-level accuracy (`p_a`) and consistency (`p_c`) described by Cui et al. (2012).
- `map_reliability`: A list with 2 elements: `accuracy` and `consistency`, which include the attribute-level classification reliability statistics described by Johnson & Sinharay (2018).
- `eap_reliability`: The attribute-level posterior probability reliability statistics described by Johnson & Sinharay (2020).

References

Cui, Y., Gierl, M. J., & Chang, H.-H. (2012). Estimating classification consistency and accuracy for cognitive diagnostic assessment. *Journal of Educational Measurement, 49*(1), 19-38. doi:10.1111/j.17453984.2011.00158.x

Johnson, M. S., & Sinharay, S. (2018). Measures of agreement to assess attribute-level classification accuracy and consistency for cognitive diagnostic assessments. *Journal of Educational Measurement, 55*(4), 635-664. doi:10.1111/jedm.12196

Johnson, M. S., & Sinharay, S. (2020). The reliability of the posterior probability of skill attainment in diagnostic classification models. *Journal of Educational and Behavioral Statistics, 45*(1), 5-31. doi:10.3102/1076998619864550

Examples

```
rstn_mdm_lcdm <- dcm_estimate(
  dcm_specify(dcmdata::mdm_qmatrix, identifier = "item"),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

reliability(rstn_mdm_lcdm)
```

score *Posterior draws of respondent proficiency*

Description

Calculate posterior draws of respondent proficiency. Optionally retain all posterior draws or return only summaries of the distribution for each respondent.

Usage

```
score(
  x,
  newdata = NULL,
  missing = NA,
  identifier = NULL,
  summary = TRUE,
  probs = c(0.025, 0.975),
  force = FALSE
)
```

Arguments

x	An estimated model (e.g., from dcm_estimate().
newdata	Optional new data. If not provided, the data used to estimate the model is scored. If provided, newdata should be a data frame with 1 row per respondent and 1 column per item. All items that appear in newdata should appear in the data used to estimate x.
missing	An R expression specifying how missing data in data is coded (e.g., NA, ".", -99, etc.). The default is NA.
identifier	Optional. Variable name of a column in newdata that contains respondent identifiers. NULL (the default) indicates that no identifiers are present in the data, and row numbers will be used as identifiers. If newdata is not specified and the data used to estimate the model is scored, the resp_id is taken from the original data.
summary	Should summary statistics be returned instead of the raw posterior draws? Only relevant if the model was estimated with a method that results in posterior distributions (e.g., "mcmc", "variational"). Default is FALSE.
probs	The percentiles to be computed by the stats::quantile() function. Only relevant if the model was estimated with a method that results in posterior distributions (e.g., "mcmc", "variational"). Only used if summary is TRUE.
force	If respondent estimates have already been added to the model object with add_respondent_estimates() should they be recalculated. Default is FALSE.

Value

A list with two elements: `class_probabilities` and `attribute_probabilities`.

If summary is `FALSE`, each element is a tibble with one row per respondent. The columns include the respondent identifier, and one column of probabilities for each of the possible classes or attributes (as `posterior::rvar()` objects).

If summary is `TRUE`, each element is a tibble with one row per respondent and class or attribute. The columns include the respondent identifier, `class` or `attribute`, `mean`, and one column for every value specified in `probs`.

Examples

```
rstn_mdm_lcdm <- dcm_estimate(
  dcm_specify(dcmdata::mdm_qmatrix, identifier = "item"),
  data = dcmdata::mdm_data,
  missing = NA,
  identifier = "respondent",
  method = "optim",
  seed = 63277,
  backend = "rstan"
)

score(rstn_mdm_lcdm, summary = FALSE)
```

---

stan-classes *S7 classes for estimation specifications*

Description

The constructors for Stan back-ends and methods are exported to support extensions to measr, for example converting other models to `measrfit` objects. We do not expect or recommend calling these functions directly unless you are converting objects, or creating new methods for measrfit objects.

Usage

```
rstan()

cmdstanr()

mcmc()

optim()

variational()

pathfinder()
```

```
gqs()
```

Details

Back-end classes:

There are two classes for estimation backends, which define the package that should be used, or was used, to estimate a model. Both classes inherit from `measr::stanbackend`.

- The `rstan()` class indicates use of the {rstan} package.
- `cmdstanr()` indicates use of the {cmdstanr} package.

Method classes:

The method classes define which estimation method should be used, or was used, for a model. All method classes inherit from `measr::stanmethod`.

- The `mcmc()` class indicates the use of Markov chain Monte Carlo via `rstan::sampling()` when using {rstan} or the `$sample()` method of the CmdStanModel class when using {cmdstanr}.
- The `variational()` class indicated the use of Stan's variational algorithm for approximate posterior sampling via `rstan::vb()` when using {rstan} or the `$variational()` method of the CmdStanModel class when using {cmdstanr}.
- The `pathfinder()` class indicates the use of pathfinder variational inference algorithm via the `$pathfinder()` method of the CmdStanModel. This method is only available when using {cmdstanr}.
- The `optim()` class indicates the use maximum-likelihood via `rstan::optimizing()` when using {rstan} or the `$optimize()` method of the CmdStanModel class when using {cmdstanr}.
- Finally, there is a `gqs()` class for use when a model has previously been estimated and were are interested in calculating generated quantities (e.g., `score()`, `loglik_array()`). The `gqs()` class indicates the use of `rstan::gqs()` when using {rstan} and the `$generate_quantities()` method of the CmdStanModel class when using {cmdstanr}.

Value

An S7 object with the corresponding class.

Examples

```
rstan()

mcmc()
```

yens_q3 *Yen's Q_3 statistic for local item dependence*

Description

Calculate the Q_3 statistic to evaluate the assumption of independent items.

Usage

```
yens_q3(x, ..., crit_value = 0.2, summary = NULL)
```

Arguments

x	A measrdcm object.
...	Unused.
crit_value	The critical value threshold for flagging the residual correlation of a given item pair. The default is 0.2, as described by Chen and Thissen (1997).
summary	A summary statistic to be returned. Must be one of "q3max" or "q3star" (see Details). If NULL (the default), no summary statistic is return, and all residual correlations are returned.

Details

Psychometric models assume that items are independent of each other, conditional on the latent trait. The Q_3 statistic (Yen, 1984) is used to evaluate this assumption. For each observed item response, we calculate the residual between the model predicted score and the observed score and then estimate correlations between the residuals across items. Each residual correlation is a Q_3 statistic.

Often, a critical values is used to flag a residual correlation above a given threshold (e.g., Chen & Thissen, 1997). Alternatively, we may use a summary statistic such as the maximum Q_3 statistic ($Q_{3,max}$; Christensen et al., 2017), or the mean-adjusted maximum Q_3 statistic ($Q_{3,*}$; Marais, 2013).

Value

If summary = NULL, a tibble with the residual correlation and flags for all item pairs. Otherwise, a numeric value representing the requested summary statistic.

References

Chen, W.-H., & Thissen, D. (1997). Local dependence indexes for item pairs using item response theory. *Journal of Educational and Behavioral Statistics, 22*(3), 265-389. doi:10.3102/10769986022003265

Christensen, K. B., Makransky, G., & Horton, M. (2017). Critical values for Yen's Q3: Identification of local dependence in the Rasch model using residual correlations. *Applied Psychological Measurement, 41*(3), 178-194. doi:10.1177/0146621616677520

Marais, I. (2013). Local dependence. In K. B. Christensen, S. Kreiner, & M. Mesbah (Eds.), *Rasch models in health* (pp. 111-130). Wiley.

Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement, 8*(2), 125-145. doi:10.1177/014662168400800201

Examples

```
model_spec <- dcm_specify(
  qmatrix = dcmdata::mdm_qmatrix,
  dentifier = "item"
)
model <- dcm_estimate(
  dcm_spec = model_spec,
  data = dcmdata::mdm_data,
  identifier = "respondent",
  method = "optim",
  seed = 63277
)

yens_q3(model)
```

Index

∗ **Bayesian**
dcm_estimate, 8
loo-waic, 14
∗ **Bayes**
bayes_factor, 4
∗ **Bradshaw**
measr_extract, 16
∗ **Chen**
yens_q3, 31
∗ **Chiu**
qmatrix_validation, 24
∗ **Cmd**
dcm_estimate, 8
∗ **DCM**
dcm_estimate, 8
∗ **Goodman**
measr_extract, 16
∗ **Model**
dcm_estimate, 8
∗ **Sinharay**
measr_extract, 16
∗ **Stan**
dcm_estimate, 8
measrdcm, 19
∗ **Templin**
measr_extract, 16
∗ **Thissen**
yens_q3, 31
∗ **Torre**
qmatrix_validation, 24
$generate_quantities(), *30*
$optimize(), *9*, *30*
$pathfinder(), *30*
$sample(), *9*, *30*
$variational(), *30*

add_criterion *(*model_evaluation*)*, 21
add_criterion(), *3*, *12*, *14*
add_fit *(*model_evaluation*)*, 21
add_fit(), *10*, *15*
add_reliability *(*model_evaluation*)*, 21
add_reliability(), *26*
add_respondent_estimates *(*model_evaluation*)*, 21
add_respondent_estimates(), *28*
aic, 3
aic(), *18*, *23*

bayes_factor, 4
bic *(*aic*)*, 3
bic(), *18*, *23*
bridgesampling::bridge_sampler(), *12*

cdi, 6
CmdStanMCMC, *20*
CmdStanModel, *9*, *30*
cmdstanr *(*stan-classes*)*, 29

dcm2::fit_m2(), *15*
dcm_estimate, 8
dcm_estimate(), *10*, *19*, *20*, *28*
dcm_specify(), *8*
draws_array, *13*

fit_m2(), *18*, *22*
fit_m2.measr::measrdcm *(*m2*)*, 15
fit_ppmc, 9, *22*
fit_ppmc(), *18*, *22*

gqs *(*stan-classes*)*, 29

log_mll, 12
log_mll(), *23*
loglik_array, 13
loglik_array(), *30*
loo, *14*
loo(), *22*
loo-waic, 14
loo.measr::measrdcm *(*loo-waic*)*, 14
loo::loo(), *18*
loo::loo.array(), *14*, *15*

loo::loo_compare(), *15*
loo::waic(), *18*
loo::waic.array(), *14, 15*
loo_compare.measr::measrdcm *(*loo-waic*)*,
14

m2, 15
mcmc *(*stan-classes*)*, 29
measr_examples, 16
measr_extract, 16
measrdcm, *3, 4*, *12–14*, 19, *21–24*, *31*
model_evaluation, 21

optim *(*stan-classes*)*, 29
options(), *8, 9*

pathfinder *(*stan-classes*)*, 29
posterior::rvar(), *29*

qmatrix_validation, 24

relative_eff(), *14*, *22*
reliability, 26
reliability(), *23*
rstan *(*stan-classes*)*, 29
rstan::gqs(), *30*
rstan::optimizing(), *9*, *30*
rstan::sampling(), *9*, *30*
rstan::stanfit, *20*
rstan::vb(), *30*

S7 object, *30*
saveRDS(), *8*
score, 28
score(), *30*
stan-classes, 29
stats::quantile(), *10*, *22*, *28*

tibble, *5*, *7*, *11*, *18*, *25*
tidyr::unnest(), *5*

variational *(*stan-classes*)*, 29

waic(), *22*
waic.measr::measrdcm *(*loo-waic*)*, 14

yens_q3, 31