Package: dcmstan (via r-universe)

June 10, 2026

Title Generate 'Stan' Code for Diagnostic Classification Models

Version 0.1.0.9000

Description Diagnostic classification models are psychometric models used to categorically estimate respondents mastery, or proficiency, on a set of predefined skills (Bradshaw, 2016, <doi:10.1002/9781118956588.ch13>). Diagnostic models can be estimated with 'Stan'; however, the necessary scripts can be long and complicated. This package automates the creation of 'Stan' scripts for diagnostic classification models. Specify different types of diagnostic models, define prior distributions, and automatically generate the necessary 'Stan' code for estimating the model.

License MIT + file LICENSE

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

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

Depends R (>= 4.3.0)

Imports cli, dagitty, dplyr, ggdag, glue, lifecycle, rdcmchecks, rlang (>= 1.1.0), S7, tibble, tidyr

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

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

Config/Needs/website r-dcm/rdcmtemplate, ggplot2

Config/Needs/documentation openpharma/roxylint

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

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE, roclets = c(``namespace'', ``rd'', ``roxylint::roxylint''))

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libglpk-dev libicu-dev libxml2-dev libssl-dev libnode-dev libx11-dev

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

Date/Publication 2025-12-02 15:57:05 UTC

RemoteUrl https://github.com/r-dcm/dcmstan

RemoteRef HEAD

RemoteSha c3297cabeb6632f762e9d41fa159fb119d441e8b

Contents

`create_profiles` *Generate mastery profiles*

Description

Given the number of attributes or model specification, generate all possible attribute patterns.

Usage

```
create_profiles(x, ...)
```

Arguments

`x`	An object used to generate the possible patterns. This could be a number (the number of attributes; e.g., `3`, `4`), or an object that defines attribute relationships (e.g., a structural model or model specification).
`...`	Additional arguments passed to methods. See details.

Details

Additional arguments passed to methods:

`keep_names`: When `x` is a model specification, should the real attribute names be used (`TRUE`; the default), or replaced with generic names (`FALSE`; e.g., `"att1"`, `"att2"`, `"att3"`).

`attributes`: When `x` is a structural model, a vector of attribute names, as in the `qmatrix_meta$attribute_names` of a DCM specification.

Value

A tibble with all possible attribute patterns. Each row is a profile, and each column indicates whether the attribute in that column was present (1) or not (0).

Examples

```
create_profiles(3L)

create_profiles(5)

create_profiles(unconstrained(), attributes = c("att1", "att2"))

create_profiles(hdcm("att1 -> att2 -> att3"),
                attributes = c("att1", "att2", "att3"))
```

---

dcm_specification *S7 model specification class*

Description

The `dcm_specification` constructor is exported to facilitate the defining of methods in other packages. We do not expect or recommend calling this function directly. Rather, to create a model specification, one should use `dcm_specify()`.

Usage

```
dcm_specification(
  qmatrix = list(),
  qmatrix_meta = list(),
  measurement_model = measurement(),
  structural_model = structural(),
  priors = dcmprior()
)
```

Arguments

qmatrix	A cleaned Q-matrix, as returned by rdcmchecks::clean_qmatrix().
qmatrix_meta	A list of Q-matrix metadata consisting of the other (not Q-matrix) elements returned by rdcmchecks::clean_qmatrix().
measurement_model	A measurement model object.
structural_model	A structural model object.
priors	A prior object.

Value

A dcm_specification object.

See Also

dcm_specify().

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)
)

dcm_specification(qmatrix = qmatrix,
                  qmatrix_meta = list(attribute_names = paste0("att", 1:4),
                                      item_identifier = NULL,
                                      item_names = 1:15),
                  measurement_model = lcdm(),
                  structural_model = unconstrained(),
                  priors = default_dcm_priors(lcdm(), unconstrained()))
```

---

dcm_specify *Specify a diagnostic classification model*

Description

Create the specifications for a Bayesian diagnostic classification model. Choose the measurement and structural models that match your assumptions of your data. Then choose your prior distributions, or use the defaults. The model specification can then be used to generate the 'Stan' code needed to estimate the model.

Usage

```
dcm_specify(
  qmatrix,
  identifier = NULL,
  measurement_model = lcdm(),
  structural_model = unconstrained(),
  priors = NULL
)
```

Arguments

qmatrix	The Q-matrix. A data frame with 1 row per item and 1 column per attribute. May optionally include an additional column of item identifiers. If an identifier column is included, this should be specified with `identifier`. All cells for the remaining attribute columns should be either 0 (item does not measure the attribute) or 1 (item does measure the attribute).
identifier	Optional. If present, the quoted name of the column in the `qmatrix` that contains item identifiers.
measurement_model	A measurement model object.
structural_model	A structural model object.
priors	A prior object created by `prior()`. If `NULL` (the default), default prior distributions defined by `default_dcm_priors()` are used.

Value

A `dcm_specification` object.

See Also

measurement-model, structural-model.

Examples

```
qmatrix <- data.frame(
  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)
)

dcm_specify(qmatrix = qmatrix,
            measurement_model = lcdm(),
            structural_model = unconstrained())
```

dcmprior *S7 prior class*

Description

The dcmprior constructor is exported to facilitate the defining of methods in other packages. We do not expect or recommend calling this function directly. Rather, to create a model specification, one should use prior() or default_dcm_priors().

Usage

```
dcmprior(
  distribution = character(0),
  type = NA_character_,
  coefficient = NA_character_,
  lower_bound = NA_real_,
  upper_bound = NA_real_
)
```

Arguments

distribution	A distribution statement for the prior (e.g., normal(0, 2)). For a complete list of available distributions, see the *Stan* documentation at https://mc-stan.org/docs/.
type	The type of parameter to apply the prior to. Parameter types will vary by model. Use get_parameters() to see list of possible types for the chosen model.
coefficient	Name of a specific parameter within the defined parameter type. If NA (the default), the prior is applied to all parameters within the type.
lower_bound	Optional. The lower bound where the distribution should be truncated.
upper_bound	Optional. The upper bound where the distribution should be truncated.

Value

A dcmprior object.

See Also

prior(), default_dcm_priors().

Examples

```
dcmprior(
  distribution = "normal(0, 1)",
  type = "intercept"
)
```

default_dcm_priors *Default priors for diagnostic classification models*

Description

View the prior distributions that are applied by default when using a given measurement and structural model.

Usage

```
default_dcm_priors(measurement_model = NULL, structural_model = NULL)
```

Arguments

measurement_model	A measurement model object.
structural_model	A structural model object.

Value

A `dcmprior` object.

Examples

```
default_dcm_priors(lcdm(), unconstrained())
default_dcm_priors(dina(), independent())
default_dcm_priors(lcdm(), loglinear())
```

generated-quantities *Generated quantities for diagnostic classification*

Description

Generated quantities are values that are calculated from model parameters, but are not directly involved in the model estimation. For example, generated quantities can be used to simulate data for posterior predictive model checks (PPMCs; e.g., Gelman et al., 2013). See details for additional information on each quantity that is available.

Usage

```
generated_quantities(loglik = FALSE, probabilities = FALSE, ppmc = FALSE)
```

Arguments

`loglik`	Logical indicating whether log-likelihood should be generated.
`probabilities`	Logical indicating whether class and attribute proficiency probabilities should be generated.
`ppmc`	Logical indicating whether replicated data sets for PPMCS should be generated.

Details

The log-likelihood contains respondent-level log-likelihood values. This may be useful when calculating relative fit indices such as the CV-LOO (Vehtari et al., 2017) or WAIC (Watanabe, 2010).

The probabilities are primary outputs of interest for respondent-level results. These quantities include the probability that each respondent belongs to each class, as well as attribute-level proficiency probabilities for each respondent.

The PPMCs generate a vector of new item responses based on the parameter values. That is, the generated quantities are replicated data sets that could be used to calculate PPMCs.

Value

A generated quantities object.

References

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian Data Analysis* (3rd ed.). Chapman & Hall/CRC. https://sites.stat.columbia.edu/gelman/book/

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. http://jmlr.org/papers/v11/watanabe10a.html

Examples

```
generated_quantities(loglik = TRUE)
```

---

get_parameters *Identify parameters included in a diagnostic classification model*

Description

When specifying prior distributions, it is often useful to see which parameters are included in a given model. Using the Q-matrix and type of diagnostic model to estimated, we can create a list of all included parameters for which a prior can be specified.

Usage

```
get_parameters(x, qmatrix, ..., identifier = NULL)
```

Arguments

x	A model specification (e.g., dcm_specify(), measurement model (e.g., lcdm()), or structural model (e.g., unconstrained()) object.
qmatrix	The Q-matrix. A data frame with 1 row per item and 1 column per attribute. May optionally include an additional column of item identifiers. If an identifier column is included, this should be specified with identifier. All cells for the remaining attribute columns should be either 0 (item does not measure the attribute) or 1 (item does measure the attribute).
...	Additional arguments passed to methods.
identifier	Optional. If present, the quoted name of the column in the qmatrix that contains item identifiers.

Value

A tibble showing the available parameter types and coefficients for a specified model.

Examples

```
qmatrix <- tibble::tibble(item = paste0("item_", 1:10),
                          att1 = sample(0:1, size = 10, replace = TRUE),
                          att2 = sample(0:1, size = 10, replace = TRUE),
                          att3 = sample(0:1, size = 10, replace = TRUE),
                          att4 = sample(0:1, size = 10, replace = TRUE))
get_parameters(dina(), qmatrix = qmatrix, identifier = "item")
```

---

measurement-model *Measurement models for diagnostic classification*

Description

The measurement model defines how the items relate to the attributes. The currently supported options for measurement models are: loglinear cognitive diagnostic model (LCDM); deterministic input, noisy "and" gate (DINA); deterministic input, noisy "or" gate (DINO); noisy-input, deterministic "and" gate (NIDA); noisy-input, deterministic "or" gate (NIDO); noncompensatory reparameterized unified model (NC-RUM); and compensatory reparameterized unified model (C-RUM). See details for additional information on each model.

Usage

```
lcdm(max_interaction = Inf)

dina()

dino()

crum()

nida()

nido()

ncrum()
```

Arguments

`max_interaction`
: For the LCDM, the highest item-level interaction to include in the model.

Details

The LCDM (Henson et al., 2009; Henson & Templin, 2019) is a general diagnostic classification model that subsumes the other more restrictive models. The probability of a respondent providing a correct response is parameterized like a regression model. For each item, the LCDM includes an intercept, which represents the log-odds of providing a correct response when none of the attributes required by the item are present. Main effect parameters represent the increase in the log-odds for each required attribute that is present. Finally, interaction terms define the change in log-odds (after the main effects) when more than one of the required attributes are present.

Non-compensatory models:

The DINA model (de la Torre & Douglas, 2004; Junker & Sijtsma, 2001) is a restrictive non-compensatory model. For each item two parameters are estimated. A guessing parameter defines the probability of a respondent providing a correct response when not all of the required attributes are present. Conversely, a slipping parameter defines the probability of providing an incorrect response when all of the required attributes are present. Thus, the DINA model takes an "all-or-nothing" approach. Either a respondent has all of the attributes required for an item, or they do not. There is no increase in the probability of providing a correct response if only a subset of the required attributes is present.

The NIDA model (Junker & Sijtsma, 2001) is a non-compensatory model that is less restrictive than the DINA model. Where the DINA model takes an "all-or-nothing" approach, the NIDA model defines the probability of responding correctly based on each attribute that has been mastered. In doing this, the NIDA model estimates parameters for each attribute and holds these parameters constant across items. Thus, respondents have increased probability of responding correctly based on the specific attributes that have been mastered. However, the parameters are held constant across items. That is, the effect of non-proficiency on an attribute is the same for all items measuring that attribute.

The reduced NC-RUM (DiBello et al., 1995; Hartz, 2002) is a non-compensatory model that is less restrictive than both the DINA and NIDA model, as the NC-RUM does not constrain

parameters across items or attributes. Thus, the NC-RUM is most similar to the LCDM; however, the equivalent LCDM parameterization of the NC-RUM constrains interaction parameters to be positive, which differs from the full LCDM specification.

Compensatory models:

The DINO model (Templin & Henson, 2006) is the inverse of the DINA model. Whereas the DINA model is "all-or-nothing", the DINO model can be thought of as "anything-or-nothing". In the DINO model, the guessing parameter defines the probability of a correct response when none of the required attributes are present. The slipping parameter is the probability of an incorrect response when any of the required attributes is present. Therefore, when using the DINO model, the presence of any of the required attributes results in an increased probability of a correct response, and there is no additional increase in probability for the presence of more than one of the required attributes.

The NIDO model (Templin, 2006) is a compensatory model that defines the probability of responding correctly based on each attribute that has been mastered. Like the NIDA model, the NIDO model holds these parameters constant across items. In the NIDO model, the probability of responding correctly increases with each mastered attribute without assuming a cumulative effect of mastering multiple attributes.

The C-RUM (Hartz, 2002) is similar to the LCDM, but is constrained to only include the intercept and main effect parameters. That is, no interaction terms are included for the C-RUM.

Value

A measurement model object.

References

de la Torre, J., & Douglas, J. A. (2004). Higher-order latent trait models for cognitive diagnosis. *Psychometrika, 69*(3), 333-353. doi:10.1007/BF02295640

DiBello, L. V., Stout, W. F., & Roussos, L. (1995). Unified cognitive psychometric assessment likelihood-based classification techniques. In P. D. Nichols, S. F. Chipman, & R. L. Brennan (Eds.), *Cognitively diagnostic assessment* (pp. 361-390). Erlbaum.

Hartz, S. M. (2002). *A Bayesian framework for the unified model for assessing cognitive abilities: Blending theory with practicality* (Publication No. 3044108) [Doctoral dissertation, University of Illinois at Urbana-Champaign]. ProQuest Dissertations Publishing.

Henson, R. A., Templin, J. L., & Willse, J. T. (2009). Defining a family of cognitive diagnosis models using log-linear models with latent variables. *Psychometrika, 74*(2), 191-210. doi:10.1007/s1133600890895

Henson, R., & Templin, J. L. (2019). Loglinear cognitive diagnostic model (LCDM). In M. von Davier & Y.-S. Lee (Eds.), *Handbook of Diagnostic Classification Models* (pp. 171-185). Springer International Publishing. doi:10.1007/9783030055844_8

Junker, B. W., & Sijtsma, K. (2001). Cognitive assessment models with few assumptions, and connections with nonparametric item response theory. *Applied Psychological Measurement, 25*(3), 258-272. doi:10.1177/01466210122032064

Templin, J. L. (2006). *CDM user's guide*. Unpublished manuscript.

Templin, J. L., & Henson, R. A. (2006). Measurement of psychological disorders using cognitive diagnosis models. *Psychological Methods, 11*(3), 287-305. doi:10.1037/1082989X.11.3.287

See Also

Structural models.

Examples

```
lcdm()

lcdm(max_interaction = 3)

dina()

dino()

nida()

nido()

ncrum()

crum()
```

`prior` *Prior definitions for diagnostic classification models*

Description

Define prior distributions for types of parameters or specific parameters within a model. For a complete list of types and parameters available for a given model, see `get_parameters()`.

Usage

```
prior(distribution, type, coefficient = NA, lower_bound = NA, upper_bound = NA)

prior_string(distribution, ...)
```

Arguments

`distribution`	A distribution statement for the prior (e.g., `normal(0, 2)`). For a complete list of available distributions, see the *Stan* documentation at https://mc-stan.org/docs/.
`type`	The type of parameter to apply the prior to. Parameter types will vary by model. Use `get_parameters()` to see list of possible types for the chosen model.
`coefficient`	Name of a specific parameter within the defined parameter type. If `NA` (the default), the prior is applied to all parameters within the type.
`lower_bound`	Optional. The lower bound where the distribution should be truncated.
`upper_bound`	Optional. The upper bound where the distribution should be truncated.
`...`	Additional arguments passed to `prior()`.

Details

prior() should be used for directly specifying priors. That is, when you are directly typing out or providing the distribution statement to the function. If you have previously created a variable with a distribution statement as a character string (e.g., dist <- "normal(0, 2)"), then you should use prior_string() to create your prior. See examples.

Value

A dcmprior object.

See Also

get_parameters().

Examples

```
prior(normal(0, 2), type = "intercept")

c(prior(beta(5, 17), type = "slip"),
  prior(beta(5, 25), type = "guess"))

my_prior <- "normal(0, 2)"
prior_string(my_prior, type = "intercept")
```

---

prior_tibble *Coerce a dcmprior object to a tibble*

Description

When specifying prior distributions, it is often useful to see which parameters are included in a given model. Using the Q-matrix and type of diagnostic model to estimated, we can create a list of all included parameters for which a prior can be specified.

Usage

```
prior_tibble(x, ...)
```

Arguments

x	A model specification (e.g., dcm_specify(), measurement model (e.g., lcdm()), or structural model (e.g., unconstrained()) object.
...	Additional arguments passed to methods. See details.

Details

Additional arguments passed to methods:

.keep_all: Logical indicating if all components should be returned. When FALSE (the default), only the @type, @coefficient, and @prior elements of the dcmprior object is return. When TRUE, the @distribtuion, @lower_bound, and @upper_bound are also returned.

Value

A tibble showing the specified priors.

Examples

```
prior_tibble(default_dcm_priors(lcdm()))

prior_tibble(default_dcm_priors(dina(), independent()))
```

`stan_code` *Generate 'Stan' code for a diagnostic classification models*

Description

Given a specification for a diagnostic classification model or a generated quantities definition, automatically generate the 'Stan' code necessary to estimate the model. For details on how the code blocks relate to diagnostic models, see da Silva et al. (2017), Jiang and Carter (2019), and Thompson (2019).

Usage

```
stan_code(x, ...)
```

Arguments

x	A model specification or generated quantities object.
...	Additional arguments passed to methods.

Value

A glue object containing the 'Stan' code for the specified model.

References

da Silva, M. A., de Oliveira, E. S. B., von Davier, A. A., and Bazán, J. L. (2017). Estimating the DINA model parameters using the No-U-Turn sampler. *Biometrical Journal, 60*(2), 352-368. doi:10.1002/bimj.201600225

Jiang, Z., & Carter, R. (2019). Using Hamiltonian Monte Carlo to estimate the log-linear cognitive diagnosis model via Stan. *Behavior Research Methods, 51*, 651-662. doi:10.3758/s1342801810699

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

```
qmatrix <- data.frame(
  att1 = sample(0:1, size = 5, replace = TRUE),
  att2 = sample(0:1, size = 5, replace = TRUE)
)

model_spec <- dcm_specify(qmatrix = qmatrix,
                          measurement_model = lcdm(),
                          structural_model = unconstrained())

stan_code(model_spec)
```

---

stan_data *Create a list of data objects for 'Stan'*

Description

When using 'Stan' to estimate a model, data objects must be passed as a list, where the name each object corresponds to the name of a variable in the `data` block of the 'Stan' code. `stan_data()` creates the list of data objects needed to estimate the model, consistent with the `data` block generated by `stan_code()`.

Usage

```
stan_data(x, ...)
```

Arguments

`x`	An object (e.g., a model specification) to create a data list for.
`...`	Additional arguments passed to methods. See details.

Details

Arguments for model specification method:

- `data`: The response data. A data frame with 1 row per respondent and 1 column per item. May optionally include an additional column of item identifiers. If an identifier is included, this should be specified with `identifier`. All cells for the remaining item columns should be either 0 (incorrect response) or 1 (correct response).
- `missing`: An expression specifying how missing values in `data` are encoded (e.g., `NA`, `"."`, `-99`). The default is `NA`.
- `identifier`: Optional. If present, the quoted name of the column in `data` that contains respondent identifiers.

Arguments for generated quantities method:

- `dcm_spec`: A cleaned data object, as returned by `rdcmchecks::clean_data()`.

- `data`: The response data. A data frame with 1 row per respondent and 1 column per item. May optionally include an additional column of item identifiers. If an identifier is included, this should be specified with `identifier`. All cells for the remaining item columns should be either 0 (incorrect response) or 1 (correct response).
- `missing`: An expression specifying how missing values in `data` are encoded (e.g., `NA`, `"."`, `-99`). The default is `NA`.
- `identifier`: Optional. If present, the quoted name of the column in `data` that contains respondent identifiers.

Value

A list of data objects.

Examples

```
qmatrix <- data.frame(
  att1 = sample(0:1, size = 5, replace = TRUE),
  att2 = sample(0:1, size = 5, replace = TRUE)
)
data <- data.frame(
  item_1 = sample(0:1, size = 20, replace = TRUE),
  item_2 = sample(0:1, size = 20, replace = TRUE),
  item_3 = sample(0:1, size = 20, replace = TRUE),
  item_4 = sample(0:1, size = 20, replace = TRUE),
  item_5 = sample(0:1, size = 20, replace = TRUE)
)

model_spec <- dcm_specify(qmatrix = qmatrix,
                          measurement_model = lcdm(),
                          structural_model = unconstrained())

stan_data(model_spec, data = data)
```

---

structural-model *Structural models for diagnostic classification*

Description

Structural models define how the attributes are related to one another. The currently supported options for structural models are: unconstrained, independent attributes, log-linear, hierarchical diagnostic classification model (HDCM), and Bayesian network. See details for additional information on each model.

Usage

```
unconstrained()

independent()
```

```
loglinear(max_interaction = Inf)

hdcm(hierarchy = NULL)

bayesnet(hierarchy = NULL)
```

Arguments

`max_interaction`	For the log-linear structural model, the highest structural-level interaction to include in the model.
`hierarchy`	Optional. If present, the quoted attribute hierarchy. See `vignette("dagitty4semusers", package = "dagitty")` for a tutorial on how to draw the attribute hierarchy.

Details

The unconstrained structural model places no constraints on how the attributes relate to each other. This is equivalent to a saturated model described by Hu & Templin (2020) and in Chapter 8 of Rupp et al. (2010).

The independent attributes model assumes that the presence of the attributes are unrelated to each other. That is, there is no relationship between the presence of one attribute and the presence of any other. For an example of independent attributes model, see Lee (2016).

The loglinear structural model assumes that class membership proportions can be estimated using a loglinear model that includes main and interaction effects (see Xu & von Davier, 2008). A saturated loglinear structural model includes interaction effects for all attributes measured in the model, and is equivalent to the unconstrained structural model and the saturated model described by Hu & Templin (2020) and in Chapter 8 of Rupp et al. (2010). A reduced form of the loglinear structural model containing only main effects is equivalent to an independent attributes model (e.g. Lee, 2016).

The hierarchical attributes model assumes some attributes must be mastered before other attributes can be mastered. For an example of the hierarchical attributes model, see Leighton et al. (2004) and Templin & Bradshaw (2014).

The Bayesian network model defines the statistical relationships between the attributes using a directed acyclic graph and a joint probability distribution. Attribute hierarchies are explicitly defined by decomposing the joint distribution for the latent attribute space into a series of marginal and conditional probability distributions. The unconstrained structural model described in Chapter 8 of Rupp et al. (2010) can be parameterized as a saturated Bayesian network (Hu & Templin, 2020). Further, structural models implying an attribute hierarchy are viewed as nested models within a saturated Bayesian network (Martinez & Templin, 2023).

Value

A structural model object.

References

Hu, B., & Templin, J. (2020). Using diagnostic classification models to validate attribute hierarchies and evaluate model fit in Bayesian Networks. *Multivariate Behavioral Research, 55*(2), 300-311. doi:10.1080/00273171.2019.1632165

Lee, S. Y. (2016). *Cognitive diagnosis model: DINA model with independent attributes*. https://mc-stan.org/learn-stan/case-studies/dina_independent.html

Leighton, J. P., Gierl, M. J., & Hunka, S. M. (2004). The attribute hierarchy method for cognitive assessment: A variation on Tatsuoka's rule-space approach. *Journal of Educational Measurement, 41*(3), 205-237. doi:10.1111/j.17453984.2004.tb01163.x

Martinez, A. J., & Templin, J. (2023). Approximate Invariance Testing in Diagnostic Classification Models in the Presence of Attribute Hierarchies: A Bayesian Network Approach. *Psych, 5*(3), 688-714. doi:10.3390/psych5030045

Rupp, A. A., Templin, J., & Henson, R. A. (2010). *Diagnostic measurement: Theory, methods, and applications*. Guilford Press.

Templin, J. L., & Bradshaw, L. (2014). Hierarchical diagnostic classification models: A family of models for estimating and testing attribute hierarchies. *Psychometrika, 79*(2), 317-339 doi:10.1007/s1133601393620

Xu, X., & von Davier, M. (2008). *Fitting the structured general diagnostic model to NAEP data* (RR-08-27). Princeton, NJ: Educational Testing Service.

See Also

Measurement models.

Examples

```
unconstrained()

independent()

loglinear()

loglinear(max_interaction = 1)

hdcm(hierarchy = "att1 -> att2 -> att3")

bayesnet(hierarchy = "att1 -> att2 -> att3")
```

Index