Package: slcm (via r-universe)

September 12, 2024

Title Sparse Latent Class Model for Cognitive Diagnosis

Version 0.1.0

Description Perform a Bayesian estimation of the exploratory Sparse Latent Class Model for Binary Data described by Chen, Y., Culpepper, S. A., and Liang, F. (2020) <doi:10.1007/s11336-019-09693-2>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.1

LinkingTo Rcpp, RcppArmadillo

Imports edmdata, Rcpp

URL `https://tmsalab.github.io/slcm/`, `https://github.com/tmsalab/slcm`

BugReports `https://github.com/tmsalab/slcm/issues`

Roxygen list(markdown = TRUE)

Suggests altdoc

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

RemoteUrl https://github.com/tmsalab/slcm

RemoteRef HEAD

RemoteSha 0bb679f2c52f5b1a11940802fd089eba8d3ea97a

Contents

attribute_pattern_table_header *Generate attribute pattern table header*

Description

Generate attribute pattern table header

Usage

```
attribute_pattern_table_header(k, m = 2, order = k)
```

Arguments

k	Number of Attributes.
m	Number of Categories. Default 2 or dichotomous response.
order	Order of the table. Default k or the full order.

Value

Return a matrix containing the class table

Examples

```
# K = 3
attribute_pattern_table_header(3)

# K = 4
attribute_pattern_table_header(4)
```

print.slcm *Print the SLCM object*

Description

Custom printing class to reveal features of the fitted SLCM.

Usage

```
## S3 method for class 'slcm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	the slcm object.
digits	the number of significant digits
...	further arguments passed to or from other methods.

Value

Print details and estimates found within the fitted SLCM. Return the model invisibly (via `invisible()`)

slcm *Sparse Latent Class Model for Cognitive Diagnosis (SLCM)*

Description

Performs the Gibbs sampling routine for a sparse latent class model as described in Chen et al. (2020) <doi: 10.1007/s11336-019-09693-2>

Usage

```
slcm(
  y,
  k,
  burnin = 1000L,
  chain_length = 10000L,
  psi_invj = c(1, rep(2, 2^k - 1)),
  m0 = 0,
  bq = 1
)
```

Arguments

`y`	Item Matrix
`k`	Dimension to estimate for Q matrix
`burnin`	Amount of Draws to Burn
`chain_length`	Number of Iterations for chain.
`psi_invj`, `m0`, `bq`	Additional tuning parameters.

Details

The `estimates` list contains the mean information from the sampling procedure. Meanwhile, the `chain` list contains full MCMC values. Lastly, the `details` list provides information regarding the estimation call.

Value

An `slcm` object containing three named lists:

- `estimates`
  - `beta`: Average beta coefficients
  - `theta`: Average theta coefficients
  - `delta`: Average activeness of coefficients

  - `class`: Average class membership
  - `pi`: Average attribute class probability.
  - `omega`: Average omega
  - `q`: Average activeness of Q matrix entries based on heuristic transformation.
  - `m2ll`: Average negative two times log-likelihood

- `chain`
  - `theta`: theta coefficients iterations
  - `beta`: beta coefficients iterations
  - `class`: class membership iterations
  - `pi`: attribute class probability iterations
  - `omega`: omega iterations
  - `m2ll`: Negative two times log-likelihood iterations
- `details`
  - `n`: Number of Subjects
  - `j`: Number of Items
  - `k`: Number of Traits
  - `l1`: Slab parameter
  - `m0`, `bq`: Additional tuning parameters
  - `burnin`: Number of Iterations to discard
  - `chain_length`: Number of Iterations to keep
  - `runtime`: Duration of model run inside of the C++ code. (Does not include summarization of MCMC chain.)
  - `package_version`: Version of the package the SLCM model was fit with.
  - `date_time`: Date and Time the model was fit.

Examples

```
# Use a demo data set from the paper
data("items_matrix_reasoning", package = "edmdata")

burnin = 50       # Set for demonstration purposes, increase to at least 1,000 in practice.
chain_length = 100 # Set for demonstration purposes, increase to at least 10,000 in practice.

model_reasoning = slcm(items_matrix_reasoning, k = 4,
                       burnin = burnin, chain_length = chain_length)

print(model_reasoning)
```

Index