Package 'tdcmStan'

March 28, 2024

Type Package

Title Automating the Creation of Stan Code for TDCMs

Version 3.0.0

Description A collection of functions for automatically creating 'Stan'
code for transition diagnostic classification models (TDCMs) as they are
defined by Madison and Bradshaw (2018) <DOI:10.1007/s11336-018-9638-5>.
This package supports automating the creation of 'Stan' code for TDCMs,
fungible TDCMs (i.e., TDCMs with item parameters constrained to be equal
across all items), and multi-threaded TDCMs.

License GPL (>= 2)

Imports dplyr (>= 1.0.7), glue (>= 1.4.2), magrittr (>= 2.0.1),
parallel (>= 4.1.0), rlang (>= 0.4.11), stringr (>= 1.4.0),
tibble (>= 3.1.5), tidyr (>= 1.1.4), tidyselect (>= 1.1.2)

Suggests readr (>= 2.0.0), testthat (>= 3.0.4)

Depends R (>= 3.5.0)

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/atlas-aai/tdcmStan

BugReports https://github.com/atlas-aai/tdcmStan/issues

NeedsCompilation no

Author Jeffrey Hoover [aut, cre, cph]
(<https://orcid.org/0000-0002-0276-0308>),
W. Jake Thompson [aut] (<https://orcid.org/0000-0001-7339-0300>)

Maintainer Jeffrey Hoover <jeffrey.c.hoover@gmail.com>

Repository CRAN

Date/Publication 2024-03-28 08:30:02 UTC

R topics documented:

---

bin_profile   *Creating a Class by Attribute Matrix*

Description

Automating the creation of Class by Attribute Matrix

Usage

```
bin_profile(natt)
```

Arguments

natt   An integer containing the number of assessed attributes.

Value

'profiles' A tibbler containing a class by attribute matrix listing which attributes are mastered by each latent class.

Examples

```
bin_profile(natt = 3)
```

---

create_fng_no_common_items_stan_tdcm
                          Creating Fungible TDCM with No Common Items Stan Code

---

Description

Automating the creation of fungible Stan code for a TDCM when there are no common items.

Usage

```
create_fng_no_common_items_stan_tdcm(q_matrix)
```

Arguments

q_matrix   A tibble containing the assessment Q-matrix.

Value

'stan_code' A list containing the text for the Stan code blocks.

Examples

```
qmatrix = tibble::tibble(att_1 = c(1, 0, 1, 0, 1, 1), att_2 = c(0, 1, 0, 1, 1, 1))
create_fng_no_common_items_stan_tdcm(q_matrix = qmatrix)
```

---

create_fng_stan_tdcm   *Creating Fungible TDCM Stan Code*

Description

Automating the creation of fungible Stan code for a TDCM.

Usage

```
create_fng_stan_tdcm(q_matrix)
```

Arguments

q_matrix   A tibble containing the assessment Q-matrix.

Value

'stan_code' A list containing the text for the Stan code blocks.

Examples

```
qmatrix = tibble::tibble(att_1 = c(1, 0, 1, 0, 1, 1), att_2 = c(0, 1, 0, 1, 1, 1))
create_fng_stan_tdcm(q_matrix = qmatrix)
```

---

create_stan_tdcm *Creating TDCM Stan Code*

Description

Automating the creation of Stan code for a TDCM.

Usage

```
create_stan_tdcm(q_matrix)
```

Arguments

q_matrix   A tibble containing the assessment Q-matrix.

Value

'stan_code' A list containing the text for the Stan code blocks.

Examples

```
qmatrix = tibble::tibble(att_1 = c(1, 0, 1, 0, 1, 1), att_2 = c(0, 1, 0, 1, 1, 1))
create_stan_tdcm(q_matrix = qmatrix)
```

---

create_threaded_stan_tdcm

Creating Multi-Threaded TDCM Stan Code

Description

Automating the creation of multi-threaded Stan code for a TDCM.

Usage

```
create_threaded_stan_tdcm(q_matrix)
```

Arguments

q_matrix   A tibble containing the assessment Q-matrix.

Value

'stan_code' A list containing the text for the Stan code blocks.

Examples

```
qmatrix = tibble::tibble(att_1 = c(1, 0, 1, 0, 1, 1), att_2 = c(0, 1, 0, 1, 1, 1))
create_threaded_stan_tdcm(q_matrix = qmatrix)
```

shard_calculator *Calculate the Number of Shards and Simultaneous Chains*

Description

Calculating the number of shards and simultaneous chains.

Usage

```
shard_calculator(num_respondents, num_responses, num_chains)
```

Arguments

num_respondents
> An integer specifying the number of respondents.

num_responses   An integer specifying the number of responses.

num_chains   An integer specifying the number of chains that need to be run.

Value

'ret' A list containing the number of shards to use within each chain and the number of chains to run in parallel.

Examples

```
shard_calculator(num_respondents = 1000, num_responses = 5000, num_chains = 4)
```

Index