Package 'rdcmchecks'

September 15, 2025

Title Common Argument Checks for 'r-dcm' Packages

Version 0.1.0

Description Many packages in the 'r-dcm' family take similar arguments, which are checked for expected structures and values. Rather than duplicating code across several packages, commonly used check functions are included here. This package can then be imported to access the check functions in other packages.

License MIT + file LICENSE

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

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

Depends R (>= 4.1.0)

Imports cli, dplyr, readr, rlang, tibble, tidyr

Suggests dcmdata, spelling, testthat (>= 3.0.0)

Config/Needs/website r-dcm/rdcmtemplate

Config/testthat/edition 3

Config/Needs/documentation openpharma/roxylint

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

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author W. Jake Thompson [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7339-0300>),
University of Kansas [cph],
Institute of Education Sciences [fnd],
Accessible Teaching, Learning, and Assessment Systems [fnd]

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

Repository CRAN

Date/Publication 2025-09-15 08:30:06 UTC

Contents

`abort_bad_argument` *Send an error message for an unexpected argument input*

Description

Send an error message for an unexpected argument input

Usage

```
abort_bad_argument(
  arg,
  must = NULL,
  not = NULL,
  footer = NULL,
  custom = NULL,
  call = rlang::caller_env()
)
```

Arguments

`arg`	The name of the argument.
`must`	The requirement for input values that is not met.
`not`	The current state of `argument` that is problematic.
`footer`	Additional text to add to the error message.
`custom`	A custom error message to override the default message of `must` + `not`.
`call`	The call stack.

Value

An error message created by `cli::cli_abort()`.

Examples

```
try(abort_bad_argument(arg = "my_arg", must = "be a character vector"))
```

check_data *Check that data follows the expected structure*

Description

The data should be 1 row per respondent and 1 column per item, with an optional additional column to store respondent identifiers. Each value of the data should be a 0 or 1 to indicate the response to the item by the given respondent. `clean_data()` calls `check_data()` to verify the expected structure, and then performs additional data manipulation to provide standard conventions. See details for additional information.

Usage

```
check_data(
  x,
  identifier = NULL,
  missing = NA,
  arg = rlang::caller_arg(x),
  call = rlang::caller_env()
)

clean_data(
  x,
  identifier = NULL,
  missing = NA,
  cleaned_qmatrix,
  arg_qmatrix = rlang::caller_arg(cleaned_qmatrix),
  valid_names = NULL,
  arg = rlang::caller_arg(x),
  call = rlang::caller_env()
)
```

Arguments

`x`	The provided data to check.
`identifier`	The provided respondent identifier, as a character string. If no respondent identifier is present, the value should be `NULL` (the default).
`missing`	A expression specifying how missing data in `x` is coded (e.g., `NA`, `"."`, `-99`). The default is `NA`.
`arg`	The name of the argument.
`call`	The call stack.
`cleaned_qmatrix`	A cleaned Q-matrix, from `clean_qmatrix()`.
`arg_qmatrix`	A character string with the name of the argument used to provide the Q-matrix.
`valid_names`	An optional named vector of items (e.g., from a previous call to `clean_data()`). Used when checking new data objects for consistency with items used to estimate a model.

Details

In many instances, it's important to have standard conventions for a data object so that we know what to expect (e.g., respondent and item identifiers, data types). `clean_data()` provides this standardization. Cleaned data is returned in long format, with one row per response. Respondent and item columns are encoded as factors, and responses are coerced to integer values.

To ensure downstream functions are able to identify the original (pre-cleaned) values, `clean_data()` returns a list that includes the cleaned data, as well as metadata that includes look-ups from the original to cleaned values.

Value

`check_data` returns the original data (if the checks pass) as a tibble, with missing data (i.e., `missing`) replaced with `NA`.

`clean_data` returns a list with five elements:

- `clean_data`: The cleaned data
- `item_identifier`: The real name of the item identifier
- `item_names`: The real names of the items
- `respondent_identifier`: The real name of the respondent identifier
- `respondent_names`: The real names of the respondents

Examples

```
example_data <- tibble::tibble(person = 1:10,
                               item1 = sample(0:1, 10, replace = TRUE),
                               item2 = sample(0:1, 10, replace = TRUE),
                               item3 = sample(0:1, 10, replace = TRUE))
check_data(example_data, identifier = "person")
example_qmatrix <- tibble::tibble(item = paste0("item", 1:3),
                                  att_1 = c(0, 0, 1),
                                  att_2 = c(1, 1, 1))

example_data <- tibble::tibble(person = 1:10,
                               item1 = sample(0:1, 10, replace = TRUE),
                               item2 = sample(0:1, 10, replace = TRUE),
                               item3 = sample(0:1, 10, replace = TRUE))

qmatrix <- clean_qmatrix(example_qmatrix, identifier = "item")
clean_data(example_data, identifier = "person",
           cleaned_qmatrix = qmatrix)
```

check_qmatrix *Check that a Q-matrix follows the expected structure*

Description

The Q-matrix should be 1 row per item and 1 column per attribute, with an optional additional column to store item identifiers. Each value of the Q-matrix should be a 0 or 1 to indicate measurement of the attribute by the given item. `clean_qmatrix()` calls `check_qmatrix()` to verify the expected structure, and then performs additional data manipulation to provide standard conventions. See details for additional information.

Usage

```
check_qmatrix(
  x,
  identifier = NULL,
  arg = rlang::caller_arg(x),
  call = rlang::caller_env()
)

clean_qmatrix(
  x,
  identifier = NULL,
  arg = rlang::caller_arg(x),
  call = rlang::caller_env()
)
```

Arguments

`x`	The provided Q-matrix to check.
`identifier`	The provided item identifier, as a character string. If no item identifier is present, the value should be `NULL` (the default).
`arg`	The name of the argument.
`call`	The call stack.

Details

In many instances, it's important to have standard conventions for a Q-matrix so that we know what to expect (e.g., item identifiers, attribute names). `clean_qmatrix()` provides this standardization. For the cleaned Q-matrix, item identifiers and item names are removed. Additionally, all attributes are renamed `att1`, `att2`, `att3`, etc. Finally, all columns are coerced to integer values.

To ensure downstream functions are able to identify the original (pre-cleaned) values, `clean_qmatrix()` returns a list that includes the cleaned Q-matrix, as well as metadata that includes look-ups from the original to cleaned values.

Value

`check_qmatrix` returns the original Q-matrix (if the checks pass) as a tibble with one row per item.

`clean_qmatrix` returns a list with four elements:

- `clean_qmatrix`: The cleaned Q-matrix
- `attribute_names`: The real names of the attributes
- `item_identifier`: The real name of the item identifier
- `item_names`: The real names of the items

Examples

```
example_qmatrix <- tibble::tibble(item = paste0("item_", 1:5),
                                  att_1 = c(0, 0, 1, 1, 1),
                                  att_2 = c(1, 1, 1, 0, 0))
check_qmatrix(example_qmatrix, identifier = "item")
example_qmatrix <- tibble::tibble(item = paste0("item_", 1:5),
                                  att_1 = c(0, 0, 1, 1, 1),
                                  att_2 = c(1, 1, 1, 0, 0))
clean_qmatrix(example_qmatrix, identifier = "item")
```

Index