

Package ‘rcoder’

October 6, 2023

Type Package

Title Lightweight Data Structure for Recoding Categorical Data without Factors

Version 0.3.0

Description A data structure and toolkit for documenting and recoding categorical data that can be shared in other statistical software.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.5)

Imports rlang, glue, dplyr

Suggests testthat (>= 2.1.0), tidyfast, magrittr

RoxygenNote 7.2.1

URL <https://github.com/nyuglobalties/rcoder>

BugReports <https://github.com/nyuglobalties/rcoder/issues>

NeedsCompilation no

Author Patrick Anker [aut, cre] (<<https://orcid.org/0000-0003-2302-0445>>),
Global TIES for Children [cph]
(<https://steinhardt.nyu.edu/ihdsc/global-ties>)

Maintainer Patrick Anker <psanker@nyu.edu>

Repository CRAN

Date/Publication 2023-10-06 08:30:05 UTC

R topics documented:

assign_coding	2
as_coding_list	3
code	3
coding	4
coding_to_haven_labels	5
coding_to_odk	5

eval_coding	6
is_empty_coding	7
link_codings	7
make_recode_query	8
matches_coding	9
missing_codes	10
odk_to_coding	10
recode_vec	11

Index	13
--------------	-----------

assign_coding	<i>Adds a coding as an attribute to a vector</i>
---------------	--------------------------------------------------

Description

Stores a coding at the "rcoder.coding" attribute of a vector

Usage

```
assign_coding(vec, .coding, .bpr = TRUE)
```

Arguments

vec	A vector
.coding	A 'coding' object
.bpr	Also overwrite the "bpr.coding" attribute with the character representation of 'coding'. Used for interop with blueprintr variable decorations.

Value

The vector with its "rcoder.coding" attribute set to 'coding'

See Also

[recode_vec()]

Examples

```
cdng <- coding(code("Yes", 3), code("Maybe", 2), code("No", 1))
vec <- sample(1:3, 50, replace = TRUE)
assign_coding(vec, cdng)
```

as_coding_list	<i>Evaluate a collection of codings from a character vector</i>
----------------	-----------------------------------------------------------------

Description

Evaluate a collection of codings from a character vector

Usage

```
as_coding_list(x)
```

Arguments

x A character vector

Value

A list of codings

Examples

```
char_vec <- c("coding(code('Yes', 1), code('No', 0))", "")
as_coding_list(char_vec)
```

code	<i>Encode a label to a value with other metadata</i>
------	------------------------------------------------------

Description

The most fundamental components of a ‘code’ object are the ‘label’ and ‘value’ elements. A ‘code’ object is essentially a key-value tuple that has some extra metadata.

Usage

```
code(
  label,
  value,
  description = label,
  links_from = label,
  missing = FALSE,
  ...
)
```

Arguments

label	A short label for a value in a vector
value	The raw value found in the respective vector
description	A longer-form label for the value, if extra context for that label is needed
links_from	A reference to another 'code' in another 'coding' object for recoding purposes.
missing	Whether this 'code' represents a missing response
...	Any extra metadata

Value

A 'code' object that contains the key-value map of label to value

Examples

```
code("Yes", 1)
code("No", 0)
code(
  "No response", -88,
  description = "Participant ignored question when prompted",
  missing = TRUE
)
code("Missing", NA, links_from = c("Refused", "Absent"))
```

coding

Catalog the categorical data representation in a vector

Description

A 'coding' object holds a list of 'code's that map vector values to human readable labels. An abstraction of factors, this data structure is designed to be portable and not directly attached to the underlying data. Moreover, 'coding' objects can be "linked" for recoding and data lineage purposes. An "empty coding" is used to represent data that has no categorical interpretation.

Usage

```
coding(..., .label = NULL)

empty_coding()
```

Arguments

...	A collection of 'code' objects
.label	A label for this coding, available for interoperability

Value

A 'coding' object that contains each 'code' input

Examples

```
coding(code("Yes", 1), code("No", 0), code("Not applicable", NA))
empty_coding()
```

`coding_to_haven_labels`*Convert coding to 'haven'-compatible labels*

Description

Converts a 'coding' object into a named vector to be used in the 'labels' parameter for 'haven::labelled()'.

Usage

```
coding_to_haven_labels(coding)
```

Arguments

`coding` A coding object

Value

A named vector representation of the coding

Examples

```
cdng <- coding(code("Yes", 1), code("No", 0))
coding_to_haven_labels(cdng)
```

`coding_to_odk`*Convert a coding object to ODK XLSForm choices*

Description

ODK XLSForms link the categorical codings to a variable type name in the 'survey' sheet. The codings are specified in the 'choices' sheet which has a 'list_name' column that holds the variable type names. Each row that has that name will be associated with that categorical type coding. This function converts 'coding' objects into tables that can be inserted into that 'choices' sheet. The categorical type is specified with the coding '.label'.

Usage

```
coding_to_odk(coding)
```

Arguments

`coding` A coding object

Value

A data.frame or tibble that can be included in an XLSForm 'choices' sheet

See Also

[odk_to_coding()]

Examples

```
cdng <- coding(code("Yes", 1), code("No", 0), .label = "yesno")
coding_to_odk(cdng)
```

eval_coding

Evaluates a coding expression in a safe environment

Description

To prevent requiring attaching the 'rcoder' package, this function takes in an unevaluated expression – assumed to be a 'coding()' call – and evaluates the expression with `_only_` 'coding' and 'code' provided to guard against rogue code.

Usage

```
eval_coding(expr)
```

Arguments

expr An expression

Value

An evaluated 'coding' object

Examples

```
eval_coding('coding(code("Yes", 1), code("No", 0))')
```

is_empty_coding	<i>Is an object the empty coding?</i>
-----------------	---------------------------------------

Description

Is an object the empty coding?

Usage

```
is_empty_coding(x)
```

Arguments

x	An object
---	-----------

Value

TRUE/FALSE if the object is identical to 'empty_coding()'

Examples

```
is_empty_coding(empty_coding())
is_empty_coding(coding())
is_empty_coding(coding(code("Yes", 1), code("No", 0)))
```

link_codings	<i>Link a coding from others for recoding</i>
--------------	-----------------------------------------------

Description

Coding objects can be linked together to create mappings from one or more codings to another. This creates a 'data.frame' that outlines how the codings are linked, to be used in 'make_recode_query'.

Usage

```
link_codings(to, ..., .to_suffix = "to", .drop_unused = FALSE)
```

Arguments

to	A coding to be linked to
...	Codings to be linked from
.to_suffix	A suffix signifying which columns in the output 'data.frame' came from 'to'
.drop_unused	Logical flag to drop any codes in '...' that have no counterparts in 'to'

Value

A 'linked_coding_df' with all necessary information for a recoding query

Examples

```

wave1 <- coding(
  code("Yes", 1),
  code("No", 2),
  code("Refused", -88, missing = TRUE)
)
wave2 <- coding(
  code("Yes", "y"),
  code("No", "n"),
  code("Missing", ".", missing = TRUE)
)
link_codings(
  to = coding(
    code("Yes", 1),
    code("No", 0),
    code("Missing", NA, links_from = c("Refused", "Missing"))
  ),
  wave1,
  wave2
)

```

make_recode_query

Make a recoding call from linked codings

Description

This creates a function that accepts a vector and recodes it from the information provided in a 'linked_coding_df'. Usually this is intended for package authors who want to operate at the recoding relational table level (e.g. mapping multiple codings to one). Most end users should use [recode_vec()] instead.

Usage

```
make_recode_query(linked_codings, from = 1, to_suffix = "to", ...)
```

Arguments

linked_codings	A 'linked_coding_df'
from	A character or integer that selects the correct original coding. Defaults to 1, the first linked coding.
to_suffix	The suffix used to signify which columns refer to values to which the vector will be recoded
...	Any other parameters passed onto the recoding function selector

Value

A function with single argument when applied to an input vector will recode the vector appropriately

Examples

```
cdng_old <- coding(code("Yes", 1), code("No", 2))
cdng_new <- coding(code("Yes", 1), code("No", 0))
recode_func <- make_recode_query(link_codings(cdng_new, cdng_old))

vec <- sample(1:2, 20, replace = TRUE)
recode_func(vec)
```

matches_coding	<i>Checks if vector's content adheres to a coding</i>
----------------	-------------------------------------------------------

Description

Performs to check to see if the set of vector values are equal to or a subset of a coding's values.

Usage

```
matches_coding(vec, coding, ignore_empty = TRUE)

verify_matches_coding(vec, coding, ignore_empty = TRUE)
```

Arguments

vec	A vector
coding	A 'coding' object
ignore_empty	Logical flag to skip check if coding is empty

Value

TRUE/FALSE

Functions

- `verify_matches_coding()`: Rather than returning TRUE/FALSE, this function halts execution if 'matches_coding()' returns FALSE.

Examples

```
vec1 <- sample(1:2, 10, replace = TRUE)
vec2 <- sample(0:1, 10, replace = TRUE)
cdng <- coding(code("Yes", 1), code("No", 0))
matches_coding(vec1, cdng)
matches_coding(vec2, cdng)
```

missing_codes	<i>Get missing codes from a coding</i>
---------------	----------------------------------------

Description

Takes a coding and returns a new coding with all codes that represent a missing value.

Usage

```
missing_codes(coding)
```

Arguments

coding a coding

Value

A coding that contains all missing codes. If no codes are found, returns 'empty_coding()'

Examples

```
missing_codes(coding(code("Yes", 1), code("No", 0), code("Missing", NA)))
missing_codes(coding(code("Yes", 1), code("No", 0)))
```

odk_to_coding	<i>Convert ODK choices to a coding</i>
---------------	----------------------------------------

Description

ODK XLSForms link the categorical codings to a variable type name in the 'survey' sheet. The codings are specified in the 'choices' sheet which has a 'list_name' column that holds the variable type names. Each row that has that name will be associated with that categorical type coding. This function converts subsets of the choices sheet into individual 'coding' objects.

Usage

```
odk_to_coding(choice_table)
```

Arguments

choice_table A data.frame slice of the "choices" table from an XLSForm

Value

A 'coding' object that corresponds to the choices' slice

See Also

[coding_to_odk()]

Examples

```
choice_excerpt <- data.frame(
  list_name = rep("yesno", 2),
  name = c("Yes", "No"),
  label = c(1, 0)
)

odk_to_coding(choice_excerpt)
```

recode_vec	<i>Recode a vector</i>
------------	------------------------

Description

A simple interface to recoding a vector based on the coding linking mechanism. If the vector has the "rcoder.coding" attribute, then the coding object stored in that attribute will be used by default.

Usage

```
recode_vec(vec, to, from = NULL, .embed = TRUE, .bpr = TRUE)
```

Arguments

<code>vec</code>	A vector
<code>to</code>	A coding object to which the vector will be recoded
<code>from</code>	A coding object that describes the current coding of the vector. Defaults to the "rcoder.coding" attribute value, if it exists, <i>or</i> the "bpr.coding" value (from blueprintr). If neither are found, 'from' stays 'NULL' and the function errors.
<code>.embed</code>	If 'TRUE', 'from' will be stored in the "rcoder.coding" attribute
<code>.bpr</code>	If 'TRUE', adds the <code>_character_</code> representation of the coding to the "bpr.coding" attribute. Used for interop with blueprintr variable decorations

Value

The recoded vector

See Also

[assign_coding()]

Examples

```
# Using an explicit `from`
vec <- sample(1:3, 50, replace = TRUE)
cdng_old <- coding(code("Yes", 3), code("Maybe", 2), code("No", 1))
cdng_new <- coding(code("Yes", 2), code("Maybe", 1), code("No", 0))
recode_vec(vec, to = cdng_new, from = cdng_old)

# Using an implicit `from` with assign_coding()
vec <- sample(1:3, 50, replace = TRUE)
vec <- assign_coding(vec, cdng_old)
recode_vec(vec, cdng_new)
```

Index

`as_coding_list`, 3
`assign_coding`, 2

`code`, 3
`coding`, 4
`coding_to_haven_labels`, 5
`coding_to_odk`, 5

`empty_coding(coding)`, 4
`eval_coding`, 6

`is_empty_coding`, 7

`link_codings`, 7

`make_recode_query`, 8
`matches_coding`, 9
`missing_codes`, 10

`odk_to_coding`, 10

`recode_vec`, 11

`verify_matches_coding(matches_coding)`,
9