Package 'composr'

December 10, 2019

Type Package

Title Compose new variables - horizontal recoding operations

Version 0.1.1

Author Martin Barner

Maintainer Martin Barner <martin.barner@impact-initiatives.org>

Description see browseVignettes(``composr''). github.com/mabafaba/composr

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat,
knitr,
rmarkdown

Imports assertthat,
crayon,
magrittr,
tibble,
glue,
koboquest (>= 1.0.0)

VignetteBuilder knitr

Remotes mabafaba/koboquest

Depends dplyr

R topics documented:

`by_sm` *apply function to each select_multiple response individually*

Description

apply function to each select_multiple response individually

Usage

```
by_sm(x, FUN, ...)
```

Arguments

`x`	a vector of select_multiple responses separated by a space " "
`FUN`	the function to be applied
`...`	further parameters passed to FUN

Value

each value in x is split into a vector on " " (space); the function in FUN is applied to each of these vectors. We return a vector of these results

---

`compose` *add layer to current composition*

Description

add layer to current composition

Usage

```
compose(.data, source, to, where.selected.any = NULL,
  where.selected.all = NULL, where.selected.exactly = NULL,
  where.selected.none = NULL, where.num.equal = NULL,
  where.num.smaller = NULL, where.num.smaller.equal = NULL,
  where.num.larger = NULL, where.num.larger.equal = NULL,
  where.string = NULL, otherwise.to = NA, skipped.to = NA,
  na.to = NA, questionnaire = NULL)
```

Arguments

`.data`	the composition, see new_composition()
`source`	the name of the source variable to compose from
`to`	the value to set the new composition to if the condition is fulfilled
`where.selected..`	: a vector of choices; setting values to 'to' where in the source variable any/all/exactly/none of the supplied choices had been selected

where.num...	: a scalar number. setting values to 'to' where the 'source' is equal / smaller / smaller or equal / larger / larger or euqal than the number supplied in where.num...
otherwise	an alternative value to be used if the condition is not fulfilled, the source is not NA and not skipped

Value

the updated composition

Examples

```
df<-data.frame(a=1:100,b=sample(letters[1:5],100,T))

df %>% new_composition("new_variable_name") %>%
compose("a",to = "less than 50" ,where.num.smaller = 50) %>%
compose("a",to = "more or equal 50", where.num.larger.equal = 50)
compose("b",to = "(size not important)",where.selected.exactly = "d") %>%
end_composition()
```

---

compose_freely *compose freely with a custom condition*

Description

compose freely with a custom condition

Usage

```
compose_freely(.data, to, where.string, questionnaire = NULL, ...)
```

Arguments

.data	an ongoing recoding (see new_recoding())
to	the value to set to
where.string	R code as a character string; evaluated in the namespace of the input data
questionnaire	if you supply a questionnaire, you will be able to use the following functions within condition: - skipped(variable_name)

end_composition *end composition*

Description

end composition

Usage

```
end_composition(.data)
```

Arguments

.data	the ongoing composition

Details

discards all composition meta information

Value

data.frame with the newly composed variable(s)

end_recoding *turn active recoding back into a simple data frame*

Description

turn active recoding back into a simple data frame

Usage

```
end_recoding(.data)
```

Arguments

.data	the recoding (see ?new_recoding)

Value

the data as a regular data.frame (tibble), with the new recoded variable added. All meta information on the recoding process is discarded.

new_composition *Start a new composition*

Description

Start a new composition

Usage

```
new_composition(df, target)
```

Arguments

df	the source data as a data.frame
target	the name of the variable that will be composed and added to the data

Value

the input data frame with - an additional column named after the value of ‘target‘ - background setup to manage step by step composition of that variable from others.

new_recoding *Start a new recoding*

Description

Start a new recoding

Usage

```
new_recoding(df, target, source = NULL)
```

Arguments

df	the source data as a data.frame
target	the name of the new variable created through the recoding
source	the variable to recode from

Details

When conditions are conflicting, the last condition that applies is used recoding is a special case of a composition, where the source variable is defined from the start and does not change.

Value

the input data frame with - an additional column named after the value of ‘target‘ - background setup to manage step by step recoding of the source variable

recode_batch *apply many recodings at once with vector of 'where' conditions*

Description

apply many recodings at once with vector of 'where' conditions

Usage

```
recode_batch(df, tos, wheres, targets = NULL, questionnaire = NULL)
```

Arguments

df	a data frame or an ongoing recoding
tos	a vector of "to" values
wheres	a vector of "where" conditions; R code as strings (evaluated in namespace of the data)
targets	vector of target variables to create as characters. each change triggers a new_recoding(). if left empty, recodes to taret specified in new_recoding().
return	the ongoing recoding from after the last 'where' recoding. return to regular data frame with all new recodings visible with end_recoding()

recode_directly *recode directly to a value*

Description

recode directly to a value

Usage

```
recode_directly(.data, to_expression, questionnaire = NULL, ...)
```

Arguments

.data	an ongoing recoding (see new_recoding())
to_expression	R code as a character string; evaluated in the namespace of the input data, and result will be the 'to' value; will overwrite everything that is not NA here
questionnaire	if you supply a questionnaire, you will be able to use 'is_skipped()' in the expression.

Details

the expression is evaluated _on each row individually_. in that world, each variable corresponds to an _individual value_. This allows you to do for example max(var1, var2) - this will return the larger value between var1 and var2 of _each record_.

recode_to *add layer to current recoding*

Description

add layer to current recoding

Usage

```
recode_to(.data, to, where.selected.any = NULL,
  where.selected.all = NULL, where.selected.exactly = NULL,
  where.selected.none = NULL, where.num.equal = NULL,
  where.num.smaller = NULL, where.num.smaller.equal = NULL,
  where.num.larger = NULL, where.num.larger.equal = NULL,
  where = NULL, otherwise.to = NA, skipped.to = NA, na.to = NA,
  questionnaire = NULL, source = NULL)
```

Arguments

.data	the ongoing recoding obejct, see new_recoding()
to	the value to set the new composition to if the condition is fulfilled
where	an R expression that will be evaluated in the namespace of the data (see example)
otherwise.to	an alternative value to be used if the condition is not fulfilled, the source is not NA and not skipped
skipped.to	an alternative value to be used if the source is NA because the question was skipped (requires to also supply the 'questionnaire' parameter)
na.to	an alternative value to be used if the source is NA but not skipped (and the condition is was not fulfilled)
source	you can set or change the source variable used; this will _continue_ to recode to the same target variable, and will continue to overwrite previously fulfilled conditions.
where.selected..	: a vector of choices; setting values to 'to' where in the source variable any/all/exactly/none of the supplied choices had been selected
where.num...	: a scalar number. setting values to 'to' where the 'source' is equal / smaller / smaller or equal / larger / larger or euqal than the number supplied in where.num...

Value

the updated recoding

Examples

```
df<-data.frame(a=1:100,b=sample(letters[1:5],100,T))

df %>%
  new_recoding("new_variable_name",a) %>%
  recode_to("less than 50" ,where.num.smaller = 50) %>%
  recode_to("more or equal 50", where.num.larger.equal = 50) %>%
```

```
 recode_to("(size not important = b equals 'd')",where.selected.exactly = "d",source = b) %>%
 end_recoding()


df %>%
  new_recoding("target_var") %>%
  recode_to(5,where = a > 3 & (b %in% letters[1:3])) %>%
  end_recoding
```

`sm_selected` *Check if select_multiple choices were selected*

Description

Check if select_multiple choices were selected

Usage

```
sm_selected(x, any = NULL, all = NULL, exactly = NULL, none = NULL)
```

Arguments

`x`	a vector of select multiple responses, with choices separated by spaces
`any`	TRUE if any of the values supplied here as a vector were selected
`all`	TRUE if all of the values supplied here as a vector were selected
`exactly`	TRUE if exactly all of the values supplied here as a vector were selected (an no others)
`any`	TRUE if none of the values supplied here as a vector were selected

Details

only supply one of any/all/exactly/any

Value

a logical vector, same length as x

Index