Package 'recode'

September 17, 2020

Type Package

Title Recodes or transforms data collected from market reserach study

Version 5.0.0

Author Kelvin Fung

Maintainer Kelvin Fung <kit319@hotmail.com>

Description This package performs various operations in data transfomation commonly seen in market reserach study

License GPL-2

Encoding UTF-8

Imports stringr, plyr, dplyr, tibble

LazyData FALSE

RoxygenNote 7.1.1

R topics documented:

back_code *Back code OTHERS into pre-coded question*

Description

This function is to back code verbatim into a pre-coded question

Usage

```
back_code(raw, coded, others_code, SN_matching)
```

Arguments

raw	dataframe with 2 columns (for backcoding with SN matching); vector (for backcoding without SN mathcing)
coded	dataframe with 2 columns (for backcoding with SN matching); vector (for backcoding without SN mathcing)
others_code	Single integer
SN_matching	Logical

Details

Inputs of this function vary upon usage; If no need for SN matching, set FALSE in the argument "SN_matching", or vice

Examples

```
raw <- data.frame(SN=c(1, 2000, 3, 4),
raw_data=c("1,2,97", "1,3", "97", "1,2,97"),
stringsAsFactors=FALSE)##Populate a dataframe
coded <- data.frame(SN=c(2000, 1, 3, 4),
coded_data=c(NA, "2,3", "9", "97"),
stringsAsFactors=FALSE)##Populate another dataframe
back_code(raw, coded, others_code = 97, SN_matching = TRUE)
#[[1]] "1,2,3"  "1,3"    "9"      "1,2,97";
#[[2]]
#SN raw_data coded_data results
#1    1   1,2,97        2,3   1,2,3
#2 2000      1,3       <NA>     1,3
#3    3       97          9       9
#4    4   1,2,97         97  1,2,97

# For matched SN
raw1 <- data.frame(raw_data=c("1,2,97", "1,3", "97", "1,2,97"),
stringsAsFactors=FALSE)##Populate a dataframe
coded1 <- data.frame(coded_data=c("5", NA, "9", "8"),
stringsAsFactors=FALSE)##Populate another dataframe
back_code(raw1$raw_data, coded1$coded_data, others_code = 97, FALSE)
#[[1]]
#[1] "1,2,5" "1,3"   "9"     "1,2,8"

#[[2]]
```

```
#raw      coded results
#[1,] "1,2,97" "5"   "1,2,5"
#[2,] "1,3"    NA    "1,3"
#[3,] "97"     "9"   "9"
#[4,] "1,2,97" "8"   "1,2,8"

# For OTHERS code other than code 97
raw2 <- data.frame(raw_data=c("1,2,91", "1,3", "91", "1,2,91"),
stringsAsFactors=FALSE)##Populate a dataframe
coded2 <- data.frame(coded_data=c("5", NA, "9", "8"),
stringsAsFactors=FALSE)##Populate another dataframe
back_code(raw2$raw_data, coded2$coded_data, others_code = 91, FALSE)
```

---

back_code_v *Back code OTHERS into pre-coded question (compatible with pipe operator)*

Description

This function is to back code verbatim into a pre-coded question

Usage

```
back_code_v(raw, coded, others_code, SN_matching)
```

Arguments

raw	dataframe with 2 columns (for backcoding with SN matching); vector (for backcoding without SN mathcing)
coded	dataframe with 2 columns (for backcoding with SN matching); vector (for backcoding without SN mathcing)
others_code	Single integer
SN_matching	Logical

Details

Inputs of this function vary upon usage; If no need for SN matching, set FALSE in the argument "SN_matching", or vice

Examples

```
##Populate a dataframe
raw <- data.frame(SN=c(1, 2000, 3, 4),
raw_data=c("1,2,97", "1,3", "97", "1,2,97"), stringsAsFactors=FALSE)
coded <- data.frame(SN=c(2000, 1, 3, 4),
coded_data=c(NA, "2,3", "9", "97"),
stringsAsFactors=FALSE)##Populate another dataframe
back_code(raw, coded, others_code = 97, SN_matching = TRUE)
#[[1]] "1,2,3"  "1,3"    "9"      "1,2,97";
#[[2]]
#SN raw_data coded_data results
#1    1   1,2,97        2,3   1,2,3
```

```
#2 2000      1,3       <NA>     1,3
#3    3       97          9       9
#4    4   1,2,97         97  1,2,97

# For matched SN
raw1 <- data.frame(raw_data=c("1,2,97", "1,3", "97", "1,2,97"),
stringsAsFactors=FALSE)##Populate a dataframe
coded1 <- data.frame(coded_data=c("5", NA, "9", "8"), stringsAsFactors=FALSE)
##Populate another dataframe
back_code(raw1$raw_data, coded1$coded_data, others_code = 97, FALSE)
#[[1]]
#[1] "1,2,5" "1,3"   "9"     "1,2,8"

#[[2]]
#raw      coded results
#[1,] "1,2,97" "5"   "1,2,5"
#[2,] "1,3"    NA    "1,3"
#[3,] "97"     "9"   "9"
#[4,] "1,2,97" "8"   "1,2,8"

# For OTHERS code other than code 97
raw2 <- data.frame(raw_data=c("1,2,91", "1,3", "91", "1,2,91"),
stringsAsFactors=FALSE)##Populate a dataframe
coded2 <- data.frame(coded_data=c("5", NA, "9", "8"),
stringsAsFactors=FALSE)##Populate another dataframe
back_code(raw2$raw_data, coded2$coded_data, others_code = 91, FALSE)
```

---

check_code *Check particular code*

Description

This function is to check if a particualr code exists

Usage

```
check_code(vector, code)
```

Arguments

vector	character vector (for MA question); numeric vector (for SA question)
code	integer

Value

logical vector

Examples

```
check_code(c("1,2,33"), 3) #FALSE
```

check_unique *Check if the variable has unique answer*

Description

This function is to check if the variable contains unique answer. Return TRUE when it does while return FALSE when it does not. For example, DK (code 98) should exist in a MA question alone; if DK (code 98) exist along with others codes in a MA question, the function will return FALSE.

Usage

```
check_unique(vector, unique_code)
```

Arguments

vector	variable to be checked
unique_code	specify unique code

Value

vector

Examples

```
check_unique(c("1,2,3", "1,2", "1"), 1)#FALSE FALSE TRUE
```

combine *Combine 2 questions together*

Description

This function is to combine 2 questions together. The returned value should have no duplicate and excessive comma.

Usage

```
combine(vector1, vector2)
```

Arguments

vector1	character vector
vector2	character vector

Examples

```
vector1 <- c("1", "2", NA, NA)
vector2 <- c("3", "2,3,4", NA, "1")
combine(vector1, vector2) #"1,3"   "2,3,4" ""      "1"
```

dup_remove *Remove duplicate(s)*

Description

This function is to remove any duplicate in MA question

Usage

```
dup_remove(vector)
```

Arguments

vector	character vector

Examples

```
dup_remove(c("1,2,3,2,3,3"))# "1,2,3"
```

dup_remove_no_message *Remove duplicate(s)*

Description

This function is to remove any duplicate in MA question

Usage

```
dup_remove_no_message(vector)
```

Arguments

vector	character vector

Examples

```
dup_remove(c("1,2,3,2,3,3"))
```

excess_comma *Remove excessive comma*

Description

This function is to remove any trailing, leading as well as consecutive comma in between.

Usage

```
excess_comma(vector)
```

Arguments

vector	character vector with excessive comma

Value

character vector without excessive comma

Examples

```
excess_comma(c(",,1,,,2,,3,,,"))# "1,2,3"
```

get_tab_var *Get variables for tabulation*

Description

This function is to get variables for tabulation

Usage

```
get_tab_var(start_with)
```

Arguments

start_with	character

Value

list

Examples

```
get_tab_var("Q")
```

`labelling` *Attach textual label to pre-coded data*

Description

This function is to attach textual label to numeric data by using a code spec

Usage

```
labelling(data, code_spec)
```

Arguments

`data`	dataframe
`code_spec`	dataframe with 3 columns: first one being "varible name", second one being "code", and thrid one being "label"

Details

This function would show output covering 3 cases: 1) The variable being processed is not included in code spec, in which case the variable would be left unchanged; 2) The variable being processed is a SA question, and 3) The variable being process is a MA question

Note that if the code is not specified in code spec, the cell containing that code would be showing NA, rather than displaying its original code

Use xlsx::write.xlsx() to export, with the argument showNA being set as FALSE.

Examples

```
data <- data.frame(vQ1=c(1, 3, 4, 99),
                   vQ2=c("1,2", "3,4", "5,3", "1,2,99"),
                   vQ3=c(NA, NA, NA, NA),
                   vQ4=c(NA, 1, 3, "2,4"),
                  vQ5=c(1,2,3,4), stringsAsFactors=FALSE)##Populate the "data" dataframe
code_spec <- data.frame(variable_name=rep(c("vQ1", "vQ2", "vQ3", "vQ4"), each=5),
                        code=rep(c(1,2,3,4,5), times=4),
                        label=rep(c("1. Ricoh",
                        "2. Fuji Xerox",
                        "3. Canon",
                        "4. Konica Minolta",
                        "5. Sharp"), times=4))##Populate the "code_spec" dataframe

labelling(data, code_spec)
```

mentions *Get the number of mentions in a MA response*

Description

This function is to Get the number of mentions in a MA response

Usage

```
mentions(vector)
```

Arguments

vector	character vector (MA question)

Examples

```
mentions(c("1", "2,4", "3,5"))#1 2 2
```

paste_dupr *Concatenate duplicate rows*

Description

This function is to concatenate duplicate rows

Usage

```
paste_dupr(id_col, concat_col, sep)
```

Arguments

id_col	character vector
concat_col	character vector
sep	character

Details

Note that this function also de-duplicate elements when concantenating.

Examples

```
#With two duplicated id; total = 29
id = c("1","2","3","4","5","5","6","7","8","9","9","10",
"11","12","13","14","15","16","17","18","19","20",
"21","22","23","24","25","26","27")
concat_col = c("4902110341812","5012427109001","8809064520019",
"7613036273800",
"9555076300000","310060143891","7613036943505","021500058506",
"078895405552","078895100020","310090143793","4898828031025",
"4901515111150","4902402534090","4899888001331","4902380188605",
"4901577042072", "4902105222843","8801043014830","4902105051306",
"4901734032083","4901085049464","4901033635053","4901033630034",
"4973344030124","4973652024501","4897020730699","4979369150106",
"8801121763933")
#Reduced to 27
paste_dupr(id, concat_col, "/")
```

---

rank_trans *Transform ranking variables*

Description

This function is to transform ranking variables to comma-separated MA question

Usage

```
rank_trans(vector, no_R)
```

Arguments

vector	character vector
no_R	single integer

Details

Argument "no_R" indicates how many attributes the respondents should rank When codes of attributes are not consecutive, e.g. attribute 4 corresponds to code 8, instead of code 4, function replace_code() can be used to replace code 4 by code 8

Examples

```
#Create a ranking dataframe
ranking_df = data.frame(attribute1 = c("3", "3", "2", "2"),
attribute2 = c("1", "1", "1", "3"),
attribute3 = c("2", "2", "4", "1"),
attribute4 = c("4", "4", "3", "4"))
#Apply the function rowwise
apply(ranking_df, 1, rank_trans, no_R = 4)
#"2,3,1,4" "2,3,1,4" "2,1,4,3" "3,1,2,4";
#the first response means rank 1, the second means rank 2 and so on
```

remove_code *Remove particular code(s)*

Description

This function is to remove code(s) in either SA or MA question

Usage

```
remove_code(vector, remove)
```

Arguments

vector	character vector (MA question) or single numeric value (SA question)
remove	character string (only 1 code to remove) or character vector (more than 1 code to remove)

Examples

```
remove_code(c("1", "2,4", "3,5"), remove=c(2,3))#remove more than 1 code in MA question; "1" "4" "5"
remove_code(c("1", "2,4", "3,5"), remove=2)#remove only 1 code in MA question; "1"  "4"  "3,5"
remove_code(c(1, 2, 3), remove=3)#remove 1 code in SA question; "1" "2" ""
```

remove_code_no_message *Remove particular code(s)*

Description

This function is to remove code(s) in either SA or MA question

Usage

```
remove_code_no_message(vector, remove)
```

Arguments

vector	character vector (MA question) or single numeric value (SA question)
remove	character string (only 1 code to remove) or character vector (more than 1 code to remove)

replace_code *Replace particular code(s)*

Description

This function is to replace code(s) in either SA or MA question

Usage

```
replace_code(vector, to_be_replaced, replacement)
```

Arguments

vector	character vector (MA question) or single numeric value (SA question)
to_be_replaced	character string (only 1 code to be replaced) or character vector (more than 1 code to be replaced)
replacement	character string (only 1 code as replacement) or character vector (more than 1 code as replacement)

Examples

```
replace_code(c("1", "2,4", "3,5"), 1, 2)#"2"   "2,4" "3,5"
replace_code(c(1, 2, 3), 1, 99)#99  2  3
```

reverse_code *Reverse codes*

Description

This function is to reverse codes in SA question

Usage

```
reverse_code(vector)
```

Arguments

vector	character vector

Details

Note that for the sake of subsequent processing of data, the output of this function is numeric

Examples

```
a = c("1", "2", "3", "4", "5", "9", NA)
reverse_code(a) #9  8  7  6  5  1 NA
```

rotate_JS *Create rotational combination in JS array format*

Description

This function is exclusively used for Rail Gen 2.0 project series to create rotational combination in JS array format

Usage

```
rotate_JS(vector = x, direction = c("left", "right"), keep_zero)
```

Arguments

vector	numeric vector
direction	c("left", "right")
keep_zero	logical

Examples

```
rotate_JS(c(1:4), "right", keep_zero=TRUE)
#"1:[0,1,2,3,4]," "2:[0,4,1,2,3]," "3:[0,3,4,1,2]," "4:[0,2,3,4,1],"
rotate_JS(c(1:4), "left", keep_zero=TRUE)
#"1:[0,1,2,3,4]," "2:[0,2,3,4,1]," "3:[0,3,4,1,2]," "4:[0,4,1,2,3],"
```

shifter *Shift elements in vector by specific distance and direction*

Description

This function is to shift elements in vector by specific distance and direction

Usage

```
shifter(vector = x, distance = n, direction = c("left", "right"))
```

Arguments

vector	numeric or character vector
distance	integer
direction	c("left", "right")

Examples

```
shifter(1:4, 0, "left")#1 2 3 4
shifter(1:4, 1, "left")#2 3 4 1
shifter(1:4, 1, "left")#Distance to be shifted exceeds length of vector
```

Index