Package ‘categorical’

December 10, 2019

Type Package

Title Representing and Manipulating categorical data in R

Version 0.1.0

Author Martin Barner <m@martinbarner.de>

Maintainer Martin Barner <m@martinbarner.de>

Description Vector types for handling categorical data.
Supports select-one and select-multiple types.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Suggests testthat (>= 2.1.0),
knitr,
rmarkdown

Imports pillar (>= 1.4.2),
purrr (>= 0.2.5),
crayon (>= 1.3.4),
tibble (>= 2.1.3),
magrittr (>= 1.5.0),
assertthat (>= 0.2.0),
dplyr (>= 0.8.3),
rlang (>= 0.4.0),
methods (>= 3.5.1),
glue (>= 1.3.0)

Depends R (>= 3.5.0),
vctrs (>= 0.2.0)

VignetteBuilder knitr

R topics documented:

`alternate` *Set categorical vector to alternative vales*

Description

Set categorical vector to alternative vales

Usage

```
alternate(x, alternative = c(), internal = FALSE)
```

Arguments

`x`	categorical vector (see [categorical()])
`alternative`	the alternative value as a string

Value

the original vector, but its active values are replaced by the alternative

---

`as_categorical` *create a new categorical variable*

Description

create a new categorical variable

Usage

```
as_categorical(x = logical(), levels = unique_and_not_na(unlist(x)),
  alternatives = empty_alternatives(levels),
  alternatives_internal = empty_alternatives(levels),
  active_alternative = NULL, active_alternative_is_internal = FALSE,
  class = c())
```

Arguments

`x`	a vector or list to be used as values for the categorical vector
`levels`	list of possible values for x; similar to factor levels
`alternatives_internal`	a named list of vectors with alternative values corresponding to 'levels'. Must have the same length as levels. Can be accessed with `categorical_alternative`. "internal" alternatives are used to store 'fixed' alternatives for classes extending 'cat_categorical'.
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `categorical_alternative`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.

`as_interval` *create a new interval variable*

Description

create a new interval variable

Usage

```
as_interval(x, levels = unique(unlist(x)), ranks = 1:length(levels),
  ...)
```

Arguments

`x`	a vector of to be used as values for the interval vector. These should be characters for most use cases (but can be other types)
`levels`	vector of of possible values for x; similar to factor levels
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `interval_alternative`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.
`rank`	a vector of numeric ranks corresponding to each level.

`as_ordinal` *create a new ordinal variable*

Description

create a new ordinal variable

Usage

```
as_ordinal(x, levels = unique(unlist(x)), ranks = 1:length(levels),
  ...)
```

Arguments

`x`	a vector of to be used as values for the ordinal vector. These should be characters for most use cases (but can be other types)
`levels`	vector of of possible values for x; similar to factor levels
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `ordinal_alternative`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.
`rank`	a vector of numeric ranks corresponding to each level.

`as_select_multiple` *convert to select_multiple variable*

Description

convert to select_multiple variable

Usage

```
as_select_multiple(x = character(), choices = NULL, labels = NULL,
  sep = " ")
```

Arguments

`x`	a character vector with concatenated select_multiple choices (for example 'c("choice_A choices_B", "choice_C")‘)
`choices`	list of options; equivalent to factor levels (in case some options were never selected but we want to track them regardless)
`labels`	named vector with choice labels. the vector name is the value in ‘x‘, the vector value is the label.
`sep`	the delimeter used to separate the choices in each element of ‘x‘ ("choice_A choice_B" vs. "choice_A; choice_B"). uses regex.

`categorical` *cat_categorical is the meta class that gives the general structure for the more specific categorical classes that extend it. cat_categorical is: - a vctrs_vctr - values are generally stored as a list (to allow select multiple and other more complex subclasses) - it has an attribute for levels / allowed values. - it has an attribute for _closed alternative values_. what these are depends on the specific sublass; for example these could be: - character labels (select) - integer rank (ordinal) - it has an attribute for _open alternative values_, allowing the user to add different alternative values, such as labels in different languages*

Description

create a new categorical variable

Usage

```
categorical(x = logical(), levels = unique_and_not_na(unlist(x)),
  alternatives = empty_alternatives(levels),
  alternatives_internal = empty_alternatives(levels),
  active_alternative = NULL, active_alternative_is_internal = FALSE,
  class = c())
```

Arguments

`x`	a vector or list to be used as values for the categorical vector
`levels`	list of possible values for x; similar to factor levels
`alternatives_internal`	a named list of vectors with alternative values corresponding to 'levels'. Must have the same length as levels. Can be accessed with `categorical_alternative`. "internal" alternatives are used to store 'fixed' alternatives for classes extending 'cat_categorical'.
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `categorical_alternative`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.

categorical.categorical

create a categorical variable from categorical input

Description

create a categorical variable from categorical input

Usage

```
## S3 method for class 'categorical'
categorical(x = logical(), levels = levels(x),
  alternatives_internal = NULL, alternatives = NULL,
  active_alternative = NULL, active_alternative_is_internal = FALSE,
  class = c())
```

categorical.matrix

categorical constructors check inputs, convert to matrix and finally should call this function.

Description

categorical constructors check inputs, convert to matrix and finally should call this function.

Usage

```
## S3 method for class 'matrix'
categorical(x = logical(), levels,
  alternatives = empty_alternatives(levels),
  alternatives_internal = empty_alternatives(levels),
  active_alternative = NULL, active_alternative_is_internal = FALSE,
  class = c())
```

categorical_logic

mutate categorical type variables, while treating each choice as logical

Description

This is much simpler than it sounds & useful; needs better description & name

Usage

```
categorical_logic(x, ...)
```

Arguments

`...`	arguments passed to dplyr::mutate
`.data`	a data.frame or tibble

Details

operates rowwise (see ?dplyr::rowwise) on a categorical column. Each row's value is a vector with the selected responses.

Value

see ?dplyr::mutate

`example_choices` *Example ODK questionnaire choices*

Description

An odk survey consists of two sheets; the questions and the choices. This is the sheet with the list of choices as a data.frame. Matching the two other example files 'example_questions' and 'example_data'.

Usage

```
example_choices
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1119 rows and 11 columns.

Source

`http://www.diamondse.info/`

`example_data` *Example data set: Needs assessment in Somalia in June 1018*

Description

The example is a small subset of the original stratified sample and should not be used in practice. Contact the package maintainer for the full dataset Matching the two other example files 'example_questions' and 'example_choices' (which are the odk xlsform as data.frames.).

Usage

```
example_data
```

Format

An object of class `list` of length 925.

Source

`http://www.diamondse.info/`

`example_questions` *Example ODK questionnaire survey*

Description

An odk survey consists of two sheets; the questions and the choices. This is the sheet with the list of questions as a data.frame. Matching the two other example files 'example_choices' and 'example_data'.

Usage

```
example_questions
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 506 rows and 14 columns.

Source

http://www.diamondse.info/

`interval` *create a new interval variable*

Description

create a new interval variable

Usage

```
interval(x, levels = unique(unlist(x)), ranks = 1:length(levels), ...)
```

Arguments

`x`	a vector of to be used as values for the interval vector. These should be characters for most use cases (but can be other types)
`levels`	vector of of possible values for x; similar to factor levels
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `alternate`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.
`rank`	a vector of numeric ranks corresponding to each level.

`is.categorical` *check if vector is of class cat_categorical*

Description

check if vector is of class cat_categorical

Usage

```
is.categorical(x)
```

Arguments

`x`	a vector

Value

TRUE if it is a categorical vector

`is.interval` *check if vector is of class cat_interval*

Description

check if vector is of class cat_interval

Usage

```
is.interval(x)
```

Arguments

`x`	a vector

Value

TRUE if it is a categorical vector

`is.ordinal` *check if vector is of class cat_ordinal*

Description

check if vector is of class cat_ordinal

Usage

```
is.ordinal(x)
```

Arguments

`x`	a vector

Value

TRUE if it is a categorical vector

`is.select_multiple` *check if vector is of class cat_select_multiple*

Description

check if vector is of class cat_select_multiple

Usage

```
is.select_multiple(x)
```

Arguments

`x`	a vector

Value

TRUE if it is

`is_categorical` *check if vector is of class cat_categorical*

Description

check if vector is of class cat_categorical

Usage

```
is_categorical(x)
```

Arguments

`x`	a vector

Value

TRUE if it is a categorical vector

`is_interval` *check if vector is of class cat_interval*

Description

check if vector is of class cat_interval

Usage

```
is_interval(x)
```

Arguments

`x`	a vector

Value

TRUE if it is a categorical vector

`is_ordinal` *check if vector is of class cat_ordinal*

Description

check if vector is of class cat_ordinal

Usage

```
is_ordinal(x)
```

Arguments

`x`	a vector

Value

TRUE if it is a categorical vector

`is_select_multiple` *check if vector is of class cat_select_multiple*

Description

check if vector is of class cat_select_multiple

Usage

```
is_select_multiple(x)
```

Arguments

`x`	a vector

Value

TRUE if it is

`label` *apply labels to categorical data*

Description

apply labels to categorical data

Usage

```
label(x)
```

Arguments

`x`	a categorical vector of class categorical_select_multiple or categorical_select_one

Value

a character vector with the labels of ‘x‘

See Also

`select_multiple`

`label.cat_select_multiple` *labels for select multiple*

Description

labels for select multiple

Usage

```
## S3 method for class 'cat_select_multiple'
label(x)
```

Arguments

`x`	a list of class categorical_select_multiple

Value

same as ‘x‘, but all values are replaced by values; lookup table for labels no longer part of attributes.

`list_alternatives` *List all alternative valuse for a categorical vector*

Description

List all alternative valuse for a categorical vector

Usage

```
list_alternatives(x, internal = NULL)
```

Arguments

`x`	a categorical vector
`internal`	logical: If TRUE, show internal alternatives only

Value

a list with internal and public alternatives as character vectors, or only one of them as a vector if ‘internal‘ is set

`mutate_categorical` *Mutate categorical type variables in a data frame*

Description

Mutate categorical type variables in a data frame

Mutate ordinal type variables in a data frame

Mutate interval type variables in a data frame

Usage

```
mutate_categorical(.data, ...)

mutate_categorical(.data, ...)

mutate_categorical(.data, ...)
```

Arguments

`.data`	a data.frame or tibble
`...`	arguments passed to dplyr::mutate
`.data`	a data.frame or tibble
`...`	arguments passed to dplyr::mutate
`.data`	a data.frame or tibble
`...`	arguments passed to dplyr::mutate

Details

operates rowwise (see ?dplyr::rowwise) on a categorical column. Each row's value is a vector with the selected responses.

operates rowwise (see ?dplyr::rowwise) on a ordinal column. Each row's value is a vector with the selected responses.

operates rowwise (see ?dplyr::rowwise) on a interval column. Each row's value is a vector with the selected responses.

Value

see ?dplyr::mutate

see ?dplyr::mutate

see ?dplyr::mutate

`mutate_select_multiple` *#' @export as.logical.cat_select_multiple<-function(x, ...) # categorical::spread_select_multiple(x)*

Description

Mutate select_multiple type variables in a data frame

Usage

```
mutate_select_multiple(.data, ...)
```

Arguments

`.data`	a data.frame or tibble
`...`	arguments passed to dplyr::mutate

Details

operates rowwise (see ?dplyr::rowwise) on a select_multiple column. Each row's value is a factor vector with the selected responses.

Value

see ?dplyr::mutate

`new_categorical` *create a new categorical variable*

Description

create a new categorical variable

Usage

```
new_categorical(x = logical(), levels,
  alternatives_internal = empty_alternatives(levels),
  alternatives = empty_alternatives(levels), active_alternative = NULL,
  active_alternative_is_internal = FALSE, class = c())
```

Arguments

`x`	a logical matrix indicating which levels are selected per record (each row is a record, each column corresponds to a level specified in 'level's)
`levels`	vector of possible values for x; similar to factor levels. Defaults to the unique values in x. Will be converted to characters
`alternatives_internal`	a named list of vectors with alternative values corresponding to 'levels'. Must have the same length as levels. Can be accessed with `categorical_alternative`. "internal" alternatives are used to store 'fixed' alternatives for classes extending 'cat_categorical'.
`alternatives`	a named list of vectors with alternative values corresponding to 'levels'. Must have the same length as levels. Can be accessed with `categorical_alternative`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.

`new_interval` *create a new interval variable*

Description

create a new interval variable

Usage

```
new_interval(lower, upper, closed = c(TRUE, FALSE), levels, levels_lower,
  levels_upper, ...)
```

Arguments

`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `alternate`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.
`x`	a vector of to be used as values for the interval vector. These should be characters for most use cases (but can be other types)
`ranks`	a vector of numeric ranks corresponding to each level.

`new_ordinal` *create a new ordinal variable*

Description

create a new ordinal variable

Usage

```
new_ordinal(x, levels, ranks, ...)
```

Arguments

`x`	a vector of to be used as values for the ordinal vector. These should be characters for most use cases (but can be other types)
`levels`	vector of of possible values for `x`; similar to factor levels
`ranks`	a vector of numeric ranks corresponding to each level.
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `alternate`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.

`new_select_multiple` *create a new select_multiple variable*

Description

create a new select_multiple variable

Usage

```
new_select_multiple(x = character(), choices = NULL, labels = NULL,
  sep = " ")
```

Arguments

`x`	a character vector with concatenated select_multiple choices (for example 'c("choice_A choices_B", "choice_C")')
`choices`	list of options; equivalent to factor levels (in case some options were never selected but we want to track them regardless)
`labels`	named vector with choice labels. the vector name is the value in 'x', the vector value is the label.
`sep`	the delimeter used to separate the choices in each element of 'x' ("choice_A choice_B" vs. "choice_A; choice_B"). uses regex.

`ordinal` *create a new ordinal variable*

Description

create a new ordinal variable

Usage

```
ordinal(x, levels = unique(unlist(x)), ranks = 1:length(levels), ...)
```

Arguments

`x`	a vector of to be used as values for the ordinal vector. These should be characters for most use cases (but can be other types)
`levels`	vector of of possible values for x; similar to factor levels
`...`	named vectors with alternative values corresponding to 'levels'. Must each have the same length as levels. Can be accessed with `alternate`. These "external" alternatives are open to user defined alternatives, for example labels in multiple languages.
`rank`	a vector of numeric ranks corresponding to each level.

`restore_lgl_list_NA_in_value_list` *set a list of items to NA where any value in a categorical logical matrix representation is NA*

Description

set a list of items to NA where any value in a categorical logical matrix representation is NA

Usage

```
restore_lgl_list_NA_in_value_list(value_list, x_categorical)
```

Arguments

`value_list`	a list with as many items as there are records in x_categorical
`x_categorical`	a categorical vector

`select_multiple` *create a select multiple variable from a character vector, list or binary matrix*

Description

create a select multiple variable from a character vector, list or binary matrix

Usage

```
select_multiple(x = character(), choices = NULL, labels = NULL,
  sep = " ")
```

Arguments

`x`	a character vector with concatenated select_multiple choices (for example 'c("choice_A choices_B", "choice_C")')
`choices`	list of options; equivalent to factor levels (in case some options were never selected but we want to track them regardless)
`labels`	named vector with choice labels. the vector name is the value in 'x', the vector value is the label.
`sep`	the delimeter used to separate the choices in each element of 'x' ("choice_A choice_B" vs. "choice_A; choice_B"). uses regex.

`superficial_nas` *find superficial NAs*

Description

find superficial NAs

Usage

```
superficial_nas(x)
```

Arguments

`x`	a <categorical> vectors

Details

"superficial NA's" appear in categorical vectors where the levels themselves are not NA, but the active alternative has no value for the level

`unique_and_not_na` *take unique values from a vector and remove all NAs*

Description

take unique values from a vector and remove all NAs

Usage

```
unique_and_not_na(x)
```

Arguments

`x`	vector

`vec_cast.cat_categorical` *cast categorical vectors*

Description

cast categorical vectors

Usage

```
## S3 method for class 'cat_categorical'
vec_cast(x, to, ...)
```

Arguments

`x`	vector
`to`	prototype to conver to
`...`	additional arguments

`vec_cast.cat_select_multiple` *vec_cast(x, to) defines the possible sets of casts. It returns x translated to have prototype to, or throws an error if the conversion isn't possible. The set of possible casts is a superset of possible coercions because they're requested explicitly. casting rules:*

Description

generally keep all all original levels and unique values as levels; keep first label for each unique level (warning if conflicting?????) select_multiple from double »> error select_multiple from integer »> error select_multiple from date »> error select_multiple from factor »> select_multiple; select_multiple from select_one »> select_multiple select_multiple from character »> select_multiple (characters as select one choices) select_one »> select_multiple

Usage

```
vec_cast.cat_select_multiple(x, to, ...)

vec_cast.cat_select_multiple(x, to, ...)
```

Arguments

`x`	vector
`to`	prototype to conver to
`...`	additional arguments

Details

select_multiple to list »> simple list; levels & labels discarded

select_multiple to atomic.. some fundamental decisions to be made.. option 1: keep as a list and cast each item using base R rules + preserves size + preserves info which response came from which record + if select_multiple allows different base types (?????), then this would be the expected behaviour in lots of cases; the goal anyway is to make select_multiples behave like regular vectors as much as possible. (maybe this line of thoughts warrants something like as_select_multiple_character().. ? –> fundamental decision whether or not to allow different base types?????) - _does not return an object of the requested class_ !

option 2: 'unlist' into single vector with all elements; then cast with base R rules. + creates a vector of the requested class - does not preserve length - does not preserve which response came from which record

option 3: require additional parameter which of the above options should be used? + makes this someone elses problem - makes this someone elses problem

option 4: throw an error + strict & somewhat makes sense; - people will _have_ to be able to convert back to basic types. - it's a massive abstraction leak; without this option the only way out is to understand how select_multiples are built internally and some basic knowledge of apply/purrr to handle

option 5: allow only as.character: concatenate everything into characters and concatenate - how to create/provide a reliable separator? - if separator is used in choices, problem - the main way forward from there is probably a strsplit, so as.character is just an less safe detour of what as.list would do

further thoughts: - select_multiples that only have one level or where at most one level is selected in each record should pose no problem. - maybe one should have to chose one of the choices with it - as.logical(x, 'option1') becomes c(TRUE, FALSE, TRUE)) - as.character(x, 'option1') becomes c("option1", "" , "option1")) - as.factor(x, 'option1') becomes factor("option1", NA, "option1")) - etc. - maybe direct coersion just shouldn't be a thing you would usually do with a select_multiple vector, and instead you would always have to take a detour: - convert to logical matrix and take it from there - collapse into single selection first and take it from there (something like forcats::fct_collapse() but considering combinations) unlike the coersions, the casting functions return _the casted object_ and not just the

---

`vec_ptype.cat_categorical`

When the change happens implicitly (e.g in c()) we call it coercion ec_ptype2(x, y) defines possible set of coercions. It returns a prototype if x and y can be safely coerced to the same prototype; otherwise it returns an error. The set of automatic coercions is usually quite small because too many tend to make code harder to reason about and silently propagate mistakes.

Description

each combo of our class and other classes that can be coerced toghether has a function that returns just a prototype of the resulting class

Usage

```
vec_ptype.cat_categorical(x)
```

Details

namespace exports: generic double dispatch boilerplate like this: #' @method vec_cast CLASS #' @export #' @export vec_cast.CLASS vec_cast.vctrs_percent <- function(x, to, ...)

individual dispatches like this:

#' @method vec_cast.CLASS1 CLASS2 #' @export

`vec_ptype2.cat_categorical`

BOILERPLATES

Description

Based on browseVignettes('vctrs')

Usage

```
## S3 method for class 'cat_categorical'
vec_ptype2(x, y, ...)
```

Arguments

`x`	vector object
`y`	vector object
`...`	additional arguments

Details

start with the boilerplate for vec_ptype2() this is just so later we can write vec_ptype2.cat_categorical.OTHERCLASS to define how these two classes should be coerced together

---

`vec_ptype2.cat_select_multiple`
When the change happens implicitly (e.g in c()) we call it coercion ec_ptype2(x, y) defines possible set of coercions. It returns a prototype if x and y can be safely coerced to the same prototype; otherwise it returns an error. The set of automatic coercions is usually quite small because too many tend to make code harder to reason about and silently propagate mistakes. each combo of our class and other classes that can be coerced toghether has a function that returns just a prototype of the resulting class start with the boilerplate for vec_ptype2() this is just so later we can write vec_ptype2.cat_select_multiple.OTHERCLASS to define how these two classes should be coerced together

---

Description

When the change happens implicitly (e.g in c()) we call it coercion ec_ptype2(x, y) defines possible set of coercions. It returns a prototype if x and y can be safely coerced to the same prototype; otherwise it returns an error. The set of automatic coercions is usually quite small because too many tend to make code harder to reason about and silently propagate mistakes. each combo of our class and other classes that can be coerced toghether has a function that returns just a prototype of the resulting class start with the boilerplate for vec_ptype2() this is just so later we can write vec_ptype2.cat_select_multiple.OTHERCLASS to define how these two classes should be coerced together

Usage

```
vec_ptype2.cat_select_multiple(x, y, ...)
```

Arguments

`x`	vector object
`y`	vector object
`...`	additional arguments

`%==exactly%` *Logical operators for select multiple*

Description

Logical operators for select multiple

Usage

```
x %==exactly% y
```

Arguments

`x`	values to check
`y`	values to check against

Index