Package 'codelist'

February 20, 2025

Type Package

Title Working with Code Lists

Version 0.1.0

Description Functions for working with code lists and vectors with codes. These are an alternative for factor that keep track of both the codes and labels. Methods allow for transforming between codes and labels. Also supports hierarchical code lists.

Depends R (>= 4.1.0)

Imports utils, methods

Suggests simplermarkdown

VignetteBuilder simplermarkdown

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Jan van der Laan [aut, cre] (<https://orcid.org/0000-0002-0693-1514>)

Maintainer Jan van der Laan <r@eoos.dds.nl>

Repository CRAN

Date/Publication 2025-02-20 18:00:06 UTC

Contents

`as.code` *Convert object to code*

Description

Convert object to code

Usage

```
as.code(x)

## S3 method for class 'code'
as.code(x)

## Default S3 method:
as.code(x)
```

Arguments

`x`	object to convert

Details

By default objects are first converted to factor using `as.factor` before being converted to code using `as.code`.

Value

Returns an object of type `code`.

`as.codelist` *Convert an object to a codelist object*

Description

Convert an object to a codelist object

Usage

```
as.codelist(x, ...)

## S3 method for class 'codelist'
as.codelist(x, ...)

## S3 method for class 'data.frame'
as.codelist(
  x,
  code = names(x)[1],
  label = names(x)[2],
  description = "description",
  parent = "parent",
  locale = "locale",
  missing = "missing",
  format = c("regular", "wide"),
  locales = NULL,
  locale_sep = "[-_@. ]",
  ...
)
```

Arguments

`x`	data.frame with the code list
`...`	used to pass extra arguments on to other methods.
`code`	the name of the column in `x` containing the codes.
`label`	the name of the column in `x` containing the labels of the codes.
`description`	the name of the column in `x` containing the labels of the codes.
`parent`	the name of the column in `x` containing the parents of the codes in case of a hierarchical code list.
`locale`	the name of the column in `x` containing the locale of the corresponding row.
`missing`	the name of the column in `x` indicating whether or not a given code should be treated as missing values.
`format`	the format of data.frame. In case of 'wide', it is assummed that columns are repeated for each locale. For example there are columns 'label_locale1' and 'label_locale2'. In case of 'regular' there are multiple rows one for each locale.

locales	only used for format = "wide". The locales in the data set.
locale_sep	the separator separating the locale from the column name. This is interpreted as a regular expression (see the 'split' argument of strsplit). The part of the column name until the first separator is the column name; the remainder the locale name.

Details

When there is no column with the name given by label in x, a new column 'label' is derived containing codes converted to character.

Value

Returns a codelist object which is a data.frame with at minimum the columns 'code' and 'label' and optionally 'description', 'parent', 'locale' and 'missing'. When x contains additional columns these are kept.

See Also

codelist for a description of the codelist object.

Examples

```
# Examples below show the same codelist in both regular and wide format
dta <- data.frame(codes = c(1:3, 1:3),
  labels = c(letters[1:3], LETTERS[1:3]),
  locale = c("en", "en", "en", "nl" ,"nl" ,"nl"))
as.codelist(dta, format = "regular")

dta <- data.frame(codes = 1:3, labels_en = letters[1:3],
  labels_nl = LETTERS[1:3])
as.codelist(dta, format = "wide")
```

---

as.label *Label character vector as label to use in comparisons with a code vector*

Description

Label character vector as label to use in comparisons with a code vector

Usage

```
as.label(x)
```

Arguments

x	character vector that is to be interpreted as a label. If x is not a character vector it will be converted to one using as.character.

Value

Returns a character vector with the class "label". This can be used in comparisons to a 'code' vector, or to assign to a 'code' vector.

See Also

Uses codes.

Examples

```
data(objectcodes)
data(objectsales)
objectsales$product <- code(objectsales$product, objectcodes)

objectsales$product[1] <- as.label("Hammer")

objectsales$product == as.label("Hammer")
subset(objectsales, product == as.label("Hammer"))

# This is the same as
subset(objectsales, product == codes("Hammer", cl(product)))
```

---

cl *Get the code list associated with the object*

Description

Get the code list associated with the object

Usage

```
cl(x)

## Default S3 method:
cl(x)

## S3 method for class 'code'
cl(x)
```

Arguments

x	the object to get the codelist from.

Value

An object of type 'codelist'.

cl_filter *Filter a code list*

Description

Filter a code list

Usage

```
cl_filter(codelist, locale, levels, check_levels = TRUE)
```

Arguments

codelist	a codelist object.
locale	use the codes from the given locale. Should be character vector of length 1. When NA the default locale is used (as returned by cl_locale.
levels	vector with levels on which to filter an hierarchical code list. Levels are numbered from 0 with 0 the topmost level. See 'Details'. When a code list does not have a 'parent' column and is, therefore, not hierarchical all codes are in level 0.
check_levels	if TRUE the parent column (if present) is removed from the result when the resulting code list would not be a valid hierarchy.

Details

When a code list has a 'parent' column. The codes without parent are assigned level 0. Codes with a parent in level 0 are assigned to level 1. Etc. When the code list does not have a 'parent' column all codes are assigned to level 0 (all codes are in the top level).

Value

Returns a codelist with the selected encoding and/or levels.

cl_is_valid *Check if the codelist is valid*

Description

Check if the codelist is valid

Usage

```
cl_is_valid(codelist)
```

Arguments

codelist	a codelist object or a data.frame that is a valid code list.

Value

Returns TRUE when the code list is valid; returns a character vector of length 1 with a description of the problem when it is not valid.

`cl_levels` *Get the hierarchical level for each code in a code list*

Description

Get the hierarchical level for each code in a code list

Usage

```
cl_levels(codelist)
```

Arguments

`codelist`	the `codelist` for which to determine the levels.

Details

Levels are numbered with 0 being the top-most level, which contains code without parent (parent missing). In level 1 are codes that have a parent in level 0. Etc.

When the code list does not have a 'parent' column, all codes are in level 0.

Value

An integer vector with the same length as the number of rows in the code list.

Examples

```
data(objectcodes)
cl_levels(objectcodes)
```

cl_locale *Get the locale to use with the codelist*

Description

Get the locale to use with the codelist

Usage

```
cl_locale(codelist, preferred = getOption("CLLOCALE", NA_character_))
```

Arguments

codelist	a codelist object or a data.frame that is a valid code list.
preferred	the preferred locale. If missing or not present in the code list, the first locale in the code list will be used.

Value

A character vector of length 1 with the locale. Can be NA when the codelist does not have locales.

cl_nlevels *Get the number of hierarchical levels in a code list*

Description

Get the number of hierarchical levels in a code list

Usage

```
cl_nlevels(codelist)
```

Arguments

codelist	the codelist for which to determine the number of levels.

Value

A single integer value (>= 1) with the number of levels.

code *Code vector*

Description

A code vector is a vector with an associated code list. The values in the vector should come from this code list. The values also have an associated label and optionally additional properties such as a description. See `codelist` for more information on what should and could be in a code list.

Usage

```
code(x, codelist, ...)
```

Arguments

`x`	vector to convert to code vector
`codelist`	code list to associate with the values in `x`. This should be convertable to `codelist` using `as.codelist`.
`...`	Ignored; used to pass extra arguments to other methods

Details

When `codelist` is omitted when case `x` is a factor, a code list is generated from the factor values.

Value

Returns an object of type 'code'. Except when `x` is a factor, `x` keeps classes and attributes assiated with `x`. This object is a copy of `x` with a `codelist` attribute added.

When `x` is a factor `x` it converted to an integer vector. The labels are the levels of the factor.

Examples

```
x <- code(c(1,4,2), codelist(codes = 1:4, labels = letters[1:4]))
print(x)
labels(x)

x <- code(factor(letters[1:3]))
print(x)
attr(x, "codelist")
```

`codelist` *Create a codelist object*

Description

Create a codelist object

Usage

```
codelist(
  codes,
  labels = NULL,
  descriptions = NULL,
  parent = NULL,
  locale = NULL,
  missing = NULL
)
```

Arguments

`codes`	a vector with the codes.
`labels`	optional vector with the labels. Will be converted to character and should have the same length as `codes`. When `labels` is not given `as.character(codes)` is used for the labels.
`descriptions`	optional vector with the descriptions of the codes. Will be converted to character and should have the same length as `codes`.
`parent`	optional vector with the parents of the codes. Should be of the same type and length as `codes` and should contain only values present in `codes` or missing values. This can be used to define simple hierarchies. Codes with NA as their parent are the top-level (level 0) codes in the hierarchy.
`locale`	optional vector with the locale of the labels, descriptions etc. of the codes. This should be a character vector with the same length as `codes`. When the code list contains multiple locales each code should be present in each locale.
`missing`	optional logical vector indicating whether or not the corresponding code can be treated as a missing value. This can be used to encode different types of missingness.

Value

Returns a `codelist` object which is a `data.frame` with at minimum the columns 'code' and 'label' and optionally 'description', 'parent', 'locale' and 'missing'. See below for a description of the columns:

`code`	The codes. It is expected that these are either characters or integers although other types are probably supported. For a given locale (see below) they should be unique. Missing values are not allowed.

label	The labels of the codes. These are characters. Missing values are not allowed.
description	Optional. The description of the codes. These are characters. Missing values are not allowed.
missing	Optional. Logical vector indicating whether or not the corresponding code can be treated as a special value. This can be used to have different codes for different types of missingness. Missing values are not allowed.
locale	Optional. Character vector indicating for the given row which locale the label and description belong to. The default use is to have different translations of the labels and descriptions. However, this can also be used, for example, to specify short and long labels. When there is more than one locale, there should be multiple lines for each code, one for each locale.
parent	Optional. The parent of the code. This can be used to specify simple hierarchies. These should be of the same type as the 'code' column and values should be present in the 'code' column or be 'NA'. When the parent is 'NA' it is assumed this is a top level code. The hierarchy should form a tree.

The validity of the code list can be checked using cl_is_valid.

---

codes *Get the codes belonging to given labels*

Description

Get the codes belonging to given labels

Usage

```
codes(x, ...)

## Default S3 method:
codes(x, codelist, locale = cl_locale(codelist), ...)

## S3 method for class 'code'
codes(x, ...)

to_codes(x, codelist, locale = cl_locale(codelist))
```

Arguments

x	character vector with labels.
...	used to pass arguments to other methods.
codelist	a codelist object or a data.frame that is a valid code list or and object that has a 'codelist' attribute containing a codelist.
locale	use the codes from the given locale. Should be a character vector of length 1.

Details

`to_codes` has the same functionality as a call to `codes.default`.

Value

Returns a vector of codes. Will give an error when one of the labels cannot be found in the codelist for the given locale. When `x` is an object of type 'code' the codes themselves are returned stripped from the 'code' class and with the 'codelist' attribute removed.

See Also

See `as.label` for an alternative in comparisons.

Examples

```
data(objectcodes)
data(objectsales)
objectsales$product <- code(objectsales$product, objectcodes)

codes(c("Hammer", "Electric Drill"), objectcodes)
codes(c("Hammer", "Electric Drill"), cl(objectsales$product))
```

`format.code` *Format a code object for pretty printing*

Description

Format a code object for pretty printing

Usage

```
## S3 method for class 'code'
format(x, maxlen = getOption("CLMAXLEN", 8L), ...)
```

Arguments

`x`	a `code` object
`maxlen`	maximum length of the label. A length of 0 or lower will suppress adding the label to the output.
`...`	ignored When `maxlen` is one or larger function will add the label of the code to the code in square brackets. When the label is larger than `maxlen` the label will be truncated.

Value

A character vector with the formatted code.

in_labels *Match codes based on label*

Description

Match codes based on label

Usage

```
in_labels(
  x,
  labels,
  codelist = attr(x, "codelist"),
  locale = cl_locale(codelist)
)
```

Arguments

x	vector with codes. Should be of the same type as the codes in the codelist.
labels	vector with labels.
codelist	a codelist object or a data.frame that is a valid code list or and object that has a 'codelist' attribute containing a codelist.
locale	use the codes from the given locale. Should be character vector of length 1.

Value

A logical vector of the same length as x indicating for each value if the code has a label present in labels.

Examples

```
data(objectcodes)
data(objectsales)
objectsales$product <- code(objectsales$product, objectcodes)

in_labels(objectsales$product, c("Electric Drill", "Toys"))

subset(objectsales, in_labels(product, c("Electric Drill", "Hammer")))
```

is.code *Check if object is a code*

Description

Check if object is a code

Usage

```
is.code(x)
```

Arguments

x	object to check

Value

Returns a logical of length 1 indicating whether or not X is of type 'code'.

is.codelist *Check if an object is a Code List*

Description

Check if an object is a Code List

Usage

```
is.codelist(x)
```

Arguments

x	object to test.

Value

Returns a logical of length 1. Returns TRUE is x is of type codelist or a data.frame that conforms to the requirements of a code list.

is.missing *Find out which elements of a vector have missing values*

Description

Find out which elements of a vector have missing values

Usage

```
is.missing(x, codelist = attr(x, "codelist"))
```

Arguments

x	vector for which the missing elements have to be detected.
codelist	a codelist object or a data.frame that is a valid code list.

Details

Unlike is.na is.missing will also return TRUE for elements of x whose values are indicated in the code list to be missing values. For that to work codelist needs to be a valid codelist with a 'missing' column. This column needs to be interpretable as a logical vector. When codelist is missing or does not contain a 'missing' column the result of is.missing is the same as is.na.

Value

Returns a logical vector of the same length as x with TRUE indicating corresponing values in x that can be considered to be missing.

labels.code *Convert vector with codes to factor using a code list*

Description

Convert vector with codes to factor using a code list

Usage

```
## S3 method for class 'code'
labels(
  object,
  missing = TRUE,
  droplevels = FALSE,
  codelist = attr(object, "codelist"),
  locale = cl_locale(codelist),
  ...
```

```
)

to_labels(
  x,
  codelist = attr(x, "codelist"),
  missing = TRUE,
  droplevels = FALSE,
  locale = cl_locale(codelist)
)
```

Arguments

object	vector with codes. Should be of the same type as the codes in the codelist.
missing	convert codes that are missing value to missing values.
droplevels	remove labels that do not occur in x.
codelist	a codelist object or a data.frame that is a valid code list.
locale	use the codes from the given locale. Should be character vector of length 1.
...	ignored
x	vector with codes. Should be of the same type as the codes in the codelist.

Details

to_labels calls labels.code directly and is meant as a substitute for labels.code for objects that are not of type 'code'.

Value

A factor vector with the same length as x.

Examples

```
data(objectsales)
data(objectcodes)
objectsales$product <- code(objectsales$product, objectcodes)

labels(objectsales$product) |>
  table(useNA = "ifany")
labels(objectsales$product, missing = FALSE) |>
  table(useNA = "ifany")
labels(objectsales$product, droplevels = TRUE) |>
  table(useNA = "ifany")

to_labels(c("A", "B"), codelist = objectcodes)
# is the same as
labels.code(c("A", "B"), codelist = objectcodes)
```

`levelcast` *Recode codes to a higher level in a hierarchy*

Description

Recode codes to a higher level in a hierarchy

Usage

```
levelcast(
  x,
  level,
  codelist = attr(x, "codelist"),
  over_level = c("error", "missing", "ignore"),
  filter_codelist = TRUE
)
```

Arguments

`x`	vector of codes to record. This can be an object of type `code`.
`level`	level to which to cast the codes.
`codelist`	the `codelist` for the codes. This code list should be hierarchical will the cast have effect.
`over_level`	how to handle codes that are in a higher level than the level that is cast to. The default 'error' will generate an error; 'missing' will result in missing values for those codes; 'ignore' will keep these codes.
`filter_codelist`	if `TRUE` codes with a level lower than the lever cast to will be removed from the code list that is returned with the result.

Details

When handling codes that are in a higher level than the level that is cast to, codes that are missing values are ignored as these are often in the highest level.

Value

A vector with the same length as `x`.

Examples

```
cl <- codelist(
    codes = c("A", "B", "A1", "A2", "B1", "B2", "A1.1", "B2.2", "X"),
    parent = c(NA, NA, "A", "A", "B", "B", "A1", "B2", NA),
    missing = c(0, 0, 0, 0, 0, 0, 0, 0, 1)
  )
x <- code(c("A1.1", "A1", "A2", "B2.2", "B2.2", NA, "B2", "X"), cl)
```

```
levelcast(x, 1)
levelcast(x, 2, over_level = "ignore")
levelcast(x, 0)
```

`objectcodes` *Example code list for object types*

Description

Contains fictional codes for various types of objects

Format

Data frame with 16 records and 5 columns.

Details

- `code` the code used for the object
- `label` label of the code
- `parent` the parent of the object in the hierarchy
- `locale` the locale of the label of the code
- `missing` should the code be treated as a missing value

`objectsales` *Example data set to demonstrate working with code lists*

Description

Contains fictional data with sales of various types of objects.

Format

Data frame with 100 records and 4 columns.

Details

- `product` the code used for the object. Corresponds to codes in `objectcodes`.
- `unitprice` price per object.
- `quantity` number of objects sold.
- `totalprice` total price of sold objects.

Index