Package: mappings (via r-universe)

September 17, 2024

Type Package

Version 1.0.0

Date 2024-06-19

Title Functions for Transforming Categorical Variables

URL https://github.com/benjaminrich/mappings

BugReports https://github.com/benjaminrich/mappings/issues

Description Easily create functions to map between different sets of values, such as for re-labeling categorical variables.

License GPL-3

Suggests testthat

Language en-US

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Repository https://benjaminrich.r-universe.dev

RemoteUrl https://github.com/benjaminrich/mappings

RemoteRef HEAD

RemoteSha 6b73295e865759d1567c31a88c989e7f7b206b85

Contents

`as.data.frame.mapping` *Convert a mapping to* `data.frame`

Description

The resulting `data.frame` has 2 columns: `mapsfrom`, and `mapsto`.

Usage

```
## S3 method for class 'mapping'
as.data.frame(x, ...)
```

Arguments

`x`	A `mapping`.
`...`	Ignored.

Value

A `data.frame`.

`cf` *Construct a* `factor` *from one or more vectors*

Description

A `factor` is constructed from one or more atomic vectors. If more than one atomic vector is supplied, then a compound value is constructed by concatenating the values together. The order of the levels is the natural order in which the values appear.

Usage

```
cf(x, ..., sep = ";")
```

Arguments

`x`	An atomic vector.
`...`	Additional atomic vectors (optional).
`sep`	A `character` to use as a separator when forming a compound value (default ';').

Value

A `factor`.

Examples

```
x <- c("A", "B", "A")
y <- c(2, 5, 7)
cf(x, y)
mapping(cf(x, y), c("X", "Y", "Z"))
```

cut_mapping *Mapping from continuous to categorical*

Description

Mapping from continuous to categorical

Usage

```
cut_mapping(..., to = NULL, na = NA, ch.as.fact = TRUE)
```

Arguments

`...`	Passed to `cut()`.
`to`	Passed to `mapping()`.
`na`	Passed to `mapping()`.
`ch.as.fact`	Passed to `mapping()`.

Value

A function that cuts a `numeric` vector and maps the result.

Examples

```
x <- c(0, 10, 20, 30, Inf)
m <- cut_mapping(x, right=FALSE,
    to=c("0 to <10", "10 to <20", "20 to <30", ">= 30"))
print(m)
m(c(5, 27, 3, 10, 99))
```

`domain` *Domain and codomain of a mapping.*

Description

Domain and codomain of a mapping.

Usage

```
domain(x)

codomain(x)
```

Arguments

`x`	A `mapping`.

Value

x A vector of the same type as we supplied when the `mapping` was created.

Note

These aren't the true domain and codomain in the mathematical sense; both can contain duplicates.

Examples

```
sex.mapping <- mapping(c("Female", "F", "Male", "M"), c(0, 0, 1, 1))
domain(sex.mapping)
codomain(sex.mapping)
```

`inverse` *Inverse of a mapping*

Description

Given a `mapping` `x`, return the inverse mapping.

Usage

```
inverse(x)
```

Arguments

`x`	A `mapping`.

Value

The inverse `mapping`.

Examples

```
sex.mapping <- mapping(c("Female", "F", "Male", "M"), c(0, 0, 1, 1))
sex.inverse.mapping <- inverse(sex.mapping)
sex.inverse.mapping(c(0, 0, 1, 0))
```

`mapping` *Generate a Mapping Function*

Description

This function returns a function that does a simple mapping from one set of value to another. It is a function-generating function.

Usage

```
mapping(from, to, na = NA, ch.as.fact = TRUE, unmapped = NA)

pmapping(..., unmapped = I)
```

Arguments

`from`	A vector. This is the domain of the function.
`to`	A vector of the same length as `from`. If omitted, then the `names` of `from` are taken as the domain, and the values as the values to map to. If `from` has no `names`, then `to` is equal to `from` (useful for re-ordering `factor` levels).
`na`	An alternative way to specify the value that `NA` maps to. Ignored if `from` contains `NA`.
`ch.as.fact`	A logical. Should the mapping return a `factor` instead of `character`?
`unmapped`	This is a fallback for the case when a value can't be mapped because it doesn't match any of the elements in `from`. It can either be a single atomic value, or a function that gets applied (which could even be another mapping). Note that this doesn't have any effect on the `inverse` mapping (which is always based solely on `from` and `to`). Default is `NA`.
`...`	Passed to `mapping()`.

Details

This function returns a function. When called with a vector argument `x`, this function will return a vector `y` of the same length as `x` and such that each element `y[i]` is equal to `to[j]` where `j` is the smallest integer such that `from[j] == x[i]`, and the value `unmapped` (or, if it's a function, `unmapped(x[i])`) if no such `j` exists.

pmapping() creates a **partial mapping**, which maps certain elements while *preserving* the rest (by making unmapped=I the default).

Note: from will always be matched as a string, even if it is numeric. So, mapping(1, "A") and mapping("1", "A") are the same, and both functions will return "A" when called with either 1 or "1".

Value

A function that translates from from to to. The function also has an inverse which is a function that performs the inverse mapping.

See Also

inverse(), codomain(), domain(), remap(), text2mapping(), cut_mapping()

Examples

```
sex.mapping <- mapping(c("Female", "F", "Male", "M"), c(0, 0, 1, 1))
sex.mapping(c("Female", "Female", "Male", "F"))

sex.mapping <- mapping(0:1, c("Female", "Male"), na="Unknown")
sex.mapping(c(0, 1, NA, 0, 1, 1, 0))
inverse(sex.mapping)(c("Female", "Male", "Unknown"))

from <- c(0, 1, NA)
to <- c(NA, "Male", "Female")
x <- c(0, 1, NA, 0, 1, 1, 0)
sex.mapping <- mapping(c(0, 1, NA), c(NA, "Male", "Female"))
sex.mapping
sex.mapping(c(0, 1, NA, 0, 1, 1, 0))
inverse(sex.mapping)
inverse(sex.mapping)(c("Female", "Male", NA))

race.mapping <- mapping(c(
      "1"="WHITE",
      "2"="BLACK OR AFRICAN AMERICAN",
      "5"="AMERICAN INDIAN OR ALASKA NATIVE"))
race.mapping(1:5)

# Use of `unmapped`
dv.mapping <- mapping("BQL", -99, unmapped=as.numeric)
dv.mapping(c("3.1", "BQL", "2.7", "100"))

# Map certain elements and preserves the rest
x <- LETTERS[1:5]
pmapping("B", "Z")(x)
mapping("B", "Z", unmapped=I)(x)  # Same
```

`paste_mapping` *A mapping that adds a prefix and/or suffix*

Description

A mapping that adds a prefix and/or suffix

Usage

```
paste_mapping(prefix = NULL, suffix = NULL)
```

Arguments

`prefix`, `suffix`	Character strings.

Value

A `mapping` function.

Examples

```
# The objective is to turn a numeric vector into a factor such that
# the levels preserve the numeric order but contain the suffix "mg"
# (i.e., so that 2 becomes "2 mg" for instance)
x <- c(1, 2, 1, 10, 3, 2, 2, 1)

# The following does not produce the levels in the desired numeric order
# (because alphabetical ordering places "10" before "2")
factor(paste(x, "mg"))

# The following works, but takes 2 lines of code and requires a variable
# assignment
y <- factor(x)
levels(y) <- paste(levels(y), "mg")
y

# This does the same thing with one line of code and no assignment
paste_mapping(, " mg")(x)

# -----

# In this example, you start with a factor, and want to preserve its ordering
x <- factor(c("Treatment", "Placebo"), levels=c("Treatment", "Placebo"))

# Again, this won't work as desired
factor(paste("Randomized to", x, "Group"))

# But this will
paste_mapping("Randomized to ", " Group")(x)
```

`print.mapping` *Print a mapping*

Description

Print a mapping

Usage

```
## S3 method for class 'mapping'
print(x, ...)
```

Arguments

`x`	`mapping`.
`...`	Ignored.

Value

Returns `x` invisibly.

`remap` *Re-map a variable*

Description

Apply a mapping to a vector directly. The mapping is temporary and not saved.

Usage

```
remap(x, ...)
```

Arguments

`x`	The values to apply the `mapping` to.
`...`	Passed to `mapping()`.

Value

The values returned by calling the `mapping` function.

Examples

```
x <- c("A", "B", "A")
remap(x, c(A=0, B=1))
```

text2mapping — *Convenient shorthand for specifying mappings with text strings*

Description

Convenient shorthand for specifying mappings with text strings

Usage

```
text2mapping(
  text,
  file = NULL,
  sep = "|",
  flip = FALSE,
  convert.na = TRUE,
  numericWherePossible = TRUE,
  ...
)
```

Arguments

`text`	A multi-line string specifying a mapping with 2 columns (see examples).
`file`	If `text` is missing, read from this file instead.
`sep`	Character used as column separator.
`flip`	If `TRUE`, flip the column order to To, From (default `FALSE`).
`convert.na`	If `TRUE`, the string `"NA"` will be converted to `NA`.
`numericWherePossible`	If `TRUE`, the mapping will return a `numeric` vector if the codomain contains only numbers.
`...`	Further arguments passed to `mapping()`.

Value

A `mapping`.

Examples

```
f <- text2mapping("
L | Low
M | Medium
H | High
")
f(warpbreaks$tension)
```

Index