Package ‘forcats’

January 29, 2023

Title  Tools for Working with Categorical Variables (Factors)

Version  1.0.0

Description  Helpers for reordering factor levels (including moving specified levels to front, ordering by first appearance, reversing, and randomly shuffling), and tools for modifying factor levels (including collapsing rare levels into other, 'anonymising', and manually 'recoding').

License  MIT + file LICENSE


BugReports  https://github.com/tidyverse/forcats/issues

Depends  R (>= 3.4)

Imports  cli (>= 3.4.0), glue, lifecycle, magrittr, rlang (>= 1.0.0), tibble

Suggests  covr, dplyr, ggplot2, knitr, readr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder  knitr

Config/Needs/website  tidyverse/tidytemplate

Config/testthat/edition  3

Encoding  UTF-8

LazyData  true

RoxygenNote  7.2.3

NeedsCompilation  no

Author  Hadley Wickham [aut, cre], RStudio [cph, fnd]

Maintainer  Hadley Wickham <hadley@rstudio.com>

Repository  CRAN

Date/Publication  2023-01-29 22:20:02 UTC
R topics documented:

- as_factor ................................................. 2
- fct .................................................. 3
- fct_anon ............................................. 4
- fct_c .................................................. 5
- fct-collapse .......................................... 6
- fct_count ............................................... 6
- fct_cross ............................................... 7
- fct_drop ............................................... 8
- fct-expand ............................................. 8
- fct-inorder ........................................... 9
- fct-lump ................................................ 10
- fct-match ............................................. 12
- fct-na-value-to-level ................................. 12
- fct-other ............................................. 13
- fct-recode ............................................ 14
- fct-relabel ........................................... 15
- fct-relevel ........................................... 16
- fct-reorder ........................................... 17
- fct-rev ................................................ 18
- fct-shift ............................................... 19
- fct-shuffle ........................................... 20
- fct-unify .............................................. 20
- fct-unique ............................................. 21
- gss_cat ................................................ 21
- lvls .................................................... 22
- lvls-union ............................................... 23

Index 24

| as_factor | Convert input to a factor |

Description

Compared to base R, when `x` is a character, this function creates levels in the order in which they appear, which will be the same on every platform. (Base R sorts in the current locale which can vary from place to place.) When `x` is numeric, the ordering is based on the numeric value and consistent with base R.

Usage

```r
as_factor(x, ...)
```

## S3 method for class 'factor'
```r
as_factor(x, ...)
```
## S3 method for class 'character'
as_factor(x, ...)

## S3 method for class 'numeric'
as_factor(x, ...)

## S3 method for class 'logical'
as_factor(x, ...)

### Arguments

- **x**: Object to coerce to a factor.
- **...**: Other arguments passed down to method.

### Details

This is a generic function.

### Examples

```r
# Character object
x <- c("a", "z", "g")
as_factor(x)
as.factor(x)

# Character object containing numbers
y <- c("1.1", "11", "2.2", "22")
as_factor(y)
as.factor(y)

# Numeric object
z <- as.numeric(y)
as_factor(z)
as.factor(z)
```

---

### fct Create a factor

**Description**

fct() is a stricter version of factor() that errors if your specification of levels is inconsistent with the values in x.

**Usage**

```r
fct(x = character(), levels = NULL, na = character())
```
Arguments

x A character vector. Values must occur in either levels or na.
levels A character vector of known levels. If not supplied, will be computed from the unique values of x, in the order in which they occur.
na A character vector of values that should become missing values.

Value

A factor.

Examples

# Use factors when you know the set of possible values a variable might take
x <- c("A", "0", "O", "AB", "A")
fct(x, levels = c("0", "A", "B", "AB"))

# If you don't specify the levels, fct will create from the data
# in the order that they're seen
fct(x)

# Differences with base R -----------------------------------------------
# factor() silently generates NAs
x <- c("a", "b", "c")
factor(x, levels = c("a", "b"))
# fct() errors
try(fct(x, levels = c("a", "b")))
# Unless you explicitly supply NA:
fct(x, levels = c("a", "b"), na = "c")

# factor() sorts default levels:
factor(c("y", "x"))
# fct() uses in order of appearance:
fct(c("y", "x"))

---

fct_anon Anonymise factor levels

Description

Replaces factor levels with arbitrary numeric identifiers. Neither the values nor the order of the levels are preserved.

Usage

fct_anon(f, prefix = "")
**Arguments**

- **f**  A factor.
- **prefix**  A character prefix to insert in front of the random labels.

**Examples**

```r
# Examples of usage

gss_cat$relig %>% fct_count()
gss_cat$relig %>%
  fct_anon() %>%
  fct_count()
gss_cat$relig %>%
  fct_anon("X") %>%
  fct_count()
```

---

**fct_c**  
*Concatenate factors, combining levels*

**Description**

This is a useful way of patching together factors from multiple sources that really should have the same levels but don’t.

**Usage**

```r
fct_c(...)  
```

**Arguments**

...  
*<dynamic-dots>*  Individual factors. Uses tidy dots, so you can splice in a list of factors with `!!!`.

**Examples**

```r
fa <- factor("a")
fb <- factor("b")
fab <- factor(c("a", "b"))

c(fa, fb, fab)
fct_c(fa, fb, fab)

# You can also pass a list of factors with `!!!`
fs <- list(fa, fb, fab)
fct_c(!!!fs)
```
fct_collapse Collapse factor levels into manually defined groups

Description

Collapse factor levels into manually defined groups

Usage

fct_collapse(.f, ..., other_level = NULL, group_other = "DEPRECATED")

Arguments

.f A factor (or character vector).
... <dynamic-dots> A series of named character vectors. The levels in each vector will be replaced with the name.
other_level Value of level used for "other" values. Always placed at end of levels.
group_other Deprecated. Replace all levels not named in ... with "Other"?

Examples

fct_count(gss_cat$partyid)

partyid2 <- fct_collapse(gss_cat$partyid,
    missing = c("No answer", "Don't know"),
    other = "Other party",
    rep = c("Strong republican", "Not str republican"),
    ind = c("Ind,near rep", "Independent", "Ind,near dem"),
    dem = c("Not str democrat", "Strong democrat")
)
fct_count(partyid2)

fct_count Count entries in a factor

Description

Count entries in a factor

Usage

fct_count(f, sort = FALSE, prop = FALSE)
**fct_cross**

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>A factor (or character vector).</td>
</tr>
<tr>
<td>sort</td>
<td>If TRUE, sort the result so that the most common values float to the top.</td>
</tr>
<tr>
<td>prop</td>
<td>If TRUE, compute the fraction of marginal table.</td>
</tr>
</tbody>
</table>

**Value**

A tibble with columns f, n and p, if prop is TRUE.

**Examples**

```r
f <- factor(sample(letters)[rpois(1000, 10)])
table(f)
fct_count(f)
fct_count(f, sort = TRUE)
fct_count(f, sort = TRUE, prop = TRUE)
```

---

**fct_cross**

*Combine levels from two or more factors to create a new factor*

**Description**

Computes a factor whose levels are all the combinations of the levels of the input factors.

**Usage**

```r
fct_cross(..., sep = ":", keep_empty = FALSE)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Additional factors or character vectors.</td>
</tr>
<tr>
<td>sep</td>
<td>A character string to separate the levels</td>
</tr>
<tr>
<td>keep_empty</td>
<td>If TRUE, keep combinations with no observations as levels</td>
</tr>
</tbody>
</table>

**Value**

The new factor

**Examples**

```r
fruit <- factor(c("apple", "kiwi", "apple", "apple"))
colour <- factor(c("green", "green", "red", "green"))
eaten <- c("yes", "no", "yes", "no")
fct_cross(fruit, colour)
fct_cross(fruit, colour, eaten)
fct_cross(fruit, colour, keep_empty = TRUE)
```
**fct_drop**

*Drop unused levels*

**Description**

Compared to `base::droplevels()`, does not drop NA levels that have values.

**Usage**

```r
fct_drop(f, only = NULL)
```

**Arguments**

- `f`: A factor (or character vector).
- `only`: A character vector restricting the set of levels to be dropped. If supplied, only levels that have no entries and appear in this vector will be removed.

**See Also**

- `fct_expand()` to add additional levels to a factor.

**Examples**

```r
f <- factor(c("a", "b"), levels = c("a", "b", "c"))
f
fct_drop(f)

# Set only to restrict which levels to drop
fct_drop(f, only = "a")
fct_drop(f, only = "c")
```

---

**fct_expand**

*Add additional levels to a factor*

**Description**

Add additional levels to a factor

**Usage**

```r
fct_expand(f, ..., after = Inf)
```

**Arguments**

- `f`: A factor (or character vector).
- `...`: Additional levels to add to the factor. Levels that already exist will be silently ignored.
- `after`: Where should the new values be placed?
fct_inorder

See Also

fct_drop() to drop unused factor levels.

Examples

```r
f <- factor(sample(letters[1:3], 20, replace = TRUE))
f
fct_expand(f, "d", "e", "f")
fct_expand(f, letters[1:6])
fct_expand(f, "Z", after = 0)
```

---

### Description

This family of functions changes only the order of the levels.

- **fct_inorder()**: by the order in which they first appear.
- **fct_infreq()**: by number of observations with each level (largest first)
- **fct_inseq()**: by numeric value of level.

### Usage

```r
fct_inorder(f, ordered = NA)
fct_infreq(f, w = NULL, ordered = NA)
fct_inseq(f, ordered = NA)
```

### Arguments

- **f**: A factor
- **ordered**: A logical which determines the "ordered" status of the output factor. NA preserves the existing status of the factor.
- **w**: An optional numeric vector giving weights for frequency of each value (not level) in f.

### Examples

```r
f <- factor(c("b", "b", "a", "c", "c", "c"))
f
fct_inorder(f)
fct_infreq(f)
```

```r
f <- factor(1:3, levels = c("3", "2", "1"))
f
fct_inseq(f)
```
**Description**

A family for lumping together levels that meet some criteria.

- `fct_lump_min(f, min, w = NULL, other_level = "Other")`: lumps levels that appear fewer than `min` times.
- `fct_lump_prop(f, prop, w = NULL, other_level = "Other")`: lumps levels that appear in fewer than (or equal to) `prop * n` times.
- `fct_lump_n(f, n, w = NULL, other_level = "Other", ties.method = c("min", "average", "first", "last", "random", "max") )`: lumps all levels except for the `n` most frequent (or least frequent if `n < 0`).
- `fct_lump_lowfreq(f, w = NULL, other_level = "Other")`: lumps together the least frequent levels, ensuring that "other" is still the smallest level.

`fct_lump()` exists primarily for historical reasons, as it automatically picks between these different methods depending on its arguments. We no longer recommend that you use it.

**Usage**

```r
fct_lump(
  f,
  n,
  prop,
  w = NULL,
  other_level = "Other",
  ties.method = c("min", "average", "first", "last", "random", "max")
)

fct_lump_min(f, min, w = NULL, other_level = "Other")

fct_lump_prop(f, prop, w = NULL, other_level = "Other")

fct_lump_n(
  f,
  n,
  w = NULL,
  other_level = "Other",
  ties.method = c("min", "average", "first", "last", "random", "max")
)

fct_lump_lowfreq(f, w = NULL, other_level = "Other")
```

**Arguments**

- `f` A factor (or character vector).
- `n` Positive `n` preserves the most common `n` values. Negative `n` preserves the least common `-n` values. It there are ties, you will get at least `abs(n)` values.
**fct_lump**

prop  Positive prop lumps values which do not appear at least prop of the time. Negative prop lumps values that do not appear at most -prop of the time.

w  An optional numeric vector giving weights for frequency of each value (not level) in f.

other_level  Value of level used for "other" values. Always placed at end of levels.

ties.method  A character string specifying how ties are treated. See `rank()` for details.

min  Preserve levels that appear at least min number of times.

**See Also**

`fct_other()` to convert specified levels to other.

**Examples**

```r
x <- factor(rep(LETTERS[1:9], times = c(40, 10, 5, 27, 1, 1, 1, 1, 1)))
x %>% table()
x %>% fct_lump_n(3) %>% table()
x %>% fct_lump_prop(0.10) %>% table()
x %>% fct_lump_min(5) %>% table()
x %>% fct_lump_lowfreq() %>% table()

x <- factor(letters[rpois(100, 5)])
x
table(x)
table(fct_lump_lowfreq(x))

# Use positive values to collapse the rarest
fct_lump_n(x, n = 3)
fct_lump_prop(x, prop = 0.1)

# Use negative values to collapse the most common
fct_lump_n(x, n = -3)
fct_lump_prop(x, prop = -0.1)

# Use weighted frequencies
w <- c(rep(2, 50), rep(1, 50))
fct_lump_n(x, n = 5, w = w)

# Use ties.method to control how tied factors are collapsed
fct_lump_n(x, n = 6)
fct_lump_n(x, n = 6, ties.method = "max")

# Use fct_lump_min() to lump together all levels with fewer than 'n' values
```
\begin{verbatim}
table(fct_lump_min(x, min = 10))
table(fct_lump_min(x, min = 15))
\end{verbatim}

\textbf{fct_match} \hspace{1cm} \textit{Test for presence of levels in a factor}

\textbf{Description}

Do any of \texttt{lvls} occur in \texttt{f}? Compared to \%in\%, this function validates \texttt{lvls} to ensure that they're actually present in \texttt{f}. In other words, \texttt{x \%in\% "not present"} will return \texttt{FALSE}, but \texttt{fct_match(x, "not present")} will throw an error.

\textbf{Usage}

\texttt{fct_match(f, lvls)}

\textbf{Arguments}

- \texttt{f} \hspace{1cm} A factor (or character vector).
- \texttt{lvls} \hspace{1cm} A character vector specifying levels to look for.

\textbf{Value}

A logical vector

\textbf{Examples}

\begin{verbatim}
  table(fct_match(gss_cat$marital, c("Married", "Divorced")))

  # Compare to %in%, misspelled levels throw an error
  table(gss_cat$marital %in% c("Maried", "Davorced"))
  ## Not run:
  table(fct_match(gss_cat$marital, c("Maried", "Davorced")))
  ## End(Not run)
\end{verbatim}

\textbf{fct_na_value_to_level} \hspace{1cm} \textit{Convert between NA values and NA levels}

\textbf{Description}

There are two ways to represent missing values in factors: in the values and in the levels. NAs in the values are most useful for data analysis (since \texttt{is.na()} returns what you expect), but because the \texttt{NA} is not explicitly recorded in the levels, there’s no way to control its position (it’s almost always displayed last or not at all). Putting the NAs in the levels allows you to control its display, at the cost of losing accurate \texttt{is.na()} reporting.

(It is possible to have a factor with missing values in both the values and the levels but it requires some explicit gymnastics and we don’t recommend it.)
Usage

\[
fct\_na\_value\_to\_level(f, \text{level} = \text{NA})
\]

\[
fct\_na\_level\_to\_value(f, \text{extra\_levels} = \text{NULL})
\]

Arguments

- \( f \) A factor (or character vector).
- \( \text{level} \) Optionally, instead of converting the NA values to an NA level, convert it to a level with this value.
- \( \text{extra\_levels} \) Optionally, a character vector giving additional levels that should also be converted to NA values.

Examples

# Most factors store NAs in the values:
\[
f1 \leftarrow \text{fct}(\text{c("a", "b", \text{NA}, "c", "b", \text{NA}))}
\]
\[
\text{levels}(f1)
\]
\[
\text{as.integer}(f1)
\]
\[
\text{is.na}(f1)
\]

# But it's also possible to store them in the levels
\[
f2 \leftarrow \text{fct\_na\_value\_to\_level}(f1)
\]
\[
\text{levels}(f2)
\]
\[
\text{as.integer}(f2)
\]
\[
\text{is.na}(f2)
\]

# If needed, you can convert back to NAs in the values:
\[
f3 \leftarrow \text{fct\_na\_level\_to\_value}(f2)
\]
\[
\text{levels}(f3)
\]
\[
\text{as.integer}(f3)
\]
\[
\text{is.na}(f3)
\]

\[\text{fct\_other} \quad \text{Manually replace levels with "other"}\]

Description

Manually replace levels with "other"

Usage

\[
fct\_other(f, \text{keep}, \text{drop}, \text{other\_level} = \text{"Other"})
\]
Arguments

- **f**: A factor (or character vector).
- **keep, drop**: Pick one of keep and drop:
  - **keep**: will preserve listed levels, replacing all others with `other_level`.
  - **drop**: will replace listed levels with `other_level`, keeping all as is.
- **other_level**: Value of level used for "other" values. Always placed at end of levels.

See Also

- `fct_lump()` to automatically convert the rarest (or most common) levels to "other".

Examples

```r
x <- factor(rep(LETTERS[1:9], times = c(40, 10, 5, 27, 1, 1, 1, 1, 1)))
fct_other(x, keep = c("A", "B"))
fct_other(x, drop = c("A", "B"))
```

---

**fct_recode**  
*Change factor levels by hand*

**Description**

Change factor levels by hand

**Usage**

```r
fct_recode(.f, ...)
```

**Arguments**

- **.f**: A factor (or character vector).
- **...**: A sequence of named character vectors where the name gives the new level, and the value gives the old level. Levels not otherwise mentioned will be left as is. Levels can be removed by naming them `NULL`.

**Examples**

```r
x <- factor(c("apple", "bear", "banana", "dear"))
fct_recode(x, fruit = "apple", fruit = "banana")

# If you make a mistake you'll get a warning
fct_recode(x, fruit = "apple", fruit = "bananana")

# If you name the level NULL it will be removed
fct_recode(x, NULL = "apple", fruit = "banana")
```
# Wrap the left hand side in quotes if it contains special variables
fct_recode(x, "an apple" = "apple", "a bear" = "bear")

# When passing a named vector to rename levels use !!! to splice
x <- factor(c("apple", "bear", "banana", "dear"))
levels <- c(fruit = "apple", fruit = "banana")
fct_recode(x, !!!levels)

---

**fct_relabel**

**Relabel factor levels with a function, collapsing as necessary**

**Description**

Relabel factor levels with a function, collapsing as necessary

**Usage**

fct_relabel(.f, .fun, ...)

**Arguments**

- **.f**
  A factor (or character vector).
- **.fun**
  A function to be applied to each level. Must accept one character argument and return a character vector of the same length as its input.
  You can also use ~ to create as shorthand (in the style of purrr). ~paste(., "x") is equivalent to function(.) paste(., "x")
- **...**
  Additional arguments to fun.

**Examples**

gss_cat$partyid %>% fct_count()
gss_cat$partyid %>%
  fct_relabel(~ gsub("", ",", .x)) %>%
  fct_count()

convert_income <- function(x) {
  regex <- "^\(?:[Ll]t\|[0-9]+\).*\$"
  is_range <- grepl(regex, x)
  num_income <- as.numeric(gsub(regex, "\1", x[is_range]))
  num_income <- trunc(num_income / 5000) * 5000
  x[is_range] <- paste0("Gt $", num_income)
  x
}
fct_count(gss_cat$rincome)
convert_income(levels(gss_cat$rincome))
rincome2 <- fct_relabel(gss_cat$rincome, convert_income)
fct_count(rincome2)
Description

This is a generalisation of `stats::relevel()` that allows you to move any number of levels to any location.

Usage

```r
fct_relevel(.f, ..., after = 0L)
```

Arguments

- `.f` A factor (or character vector).
- `...` Either a function (or formula), or character levels.
  
  A function will be called with the current levels as input, and the return value (which must be a character vector) will be used to relevel the factor. Any levels not mentioned will be left in their existing order, by default after the explicitly mentioned levels. Supports tidy dots.
- `after` Where should the new values be placed?

Examples

```r
f <- factor(c("a", "b", "c", "d"), levels = c("b", "c", "d", "a"))
fct_relevel(f)
fct_relevel(f, "a")
fct_relevel(f, "b", "a")

# Move to the third position
fct_relevel(f, "a", after = 2)

# Relevel to the end
fct_relevel(f, "a", after = Inf)
fct_relevel(f, "a", after = 3)

# Relevel with a function
fct_relevel(f, sort)
fct_relevel(f, sample)
fct_relevel(f, rev)

# Using 'Inf' allows you to relevel to the end when the number
# of levels is unknown or variable (e.g. vectorised operations)
df <- forcats::gss_cat[, c("rincome", "denom")]
lapply(df, levels)
df2 <- lapply(df, fct_relevel, "Don't know", after = Inf)
lapply(df2, levels)
```
You'll get a warning if the levels don't exist
`fct_relevel(f, "e")`

---

**fct_reorder**

*Reorder factor levels by sorting along another variable*

**Description**

`fct_reorder()` is useful for 1d displays where the factor is mapped to position; `fct_reorder2()` for 2d displays where the factor is mapped to a non-position aesthetic. `last2()` and `first2()` are helpers for `fct_reorder2()`: `last2()` finds the last value of y when sorted by x; `first2()` finds the first value.

**Usage**

```r
fct_reorder(
  .f,
  .x,
  .fun = median,
  ..., 
  .na.rm = NULL,
  .default = Inf,
  .desc = FALSE
)
```

```r
fct_reorder2(
  .f,
  .x,
  .y,
  .fun = last2,
  ..., 
  .na.rm = NULL,
  .default = -Inf,
  .desc = TRUE
)
```

`last2(.x, .y)`

`first2(.x, .y)`

**Arguments**

- `.f` A factor (or character vector).
- `.x, .y` The levels of `f` are reordered so that the values of `.fun(.x)` (for `fct_reorder()`) and `.fun(.x, .y)` (for `fct_reorder2()`) are in ascending order.
Reverse order of factor levels

Description

This is sometimes useful when plotting a factor.
Usage

`fct_rev(f)`

Arguments

- `f`  A factor (or character vector).

Examples

```r
f <- factor(c("a", "b", "c"))
fct_rev(f)
```

---

**fct_shift**  
*Shift factor levels to left or right, wrapping around at end*

Description

This is useful when the levels of an ordered factor are actually cyclical, with different conventions on the starting point.

Usage

`fct_shift(f, n = 1L)`

Arguments

- `f`  A factor.
- `n`  Positive values shift to the left; negative values shift to the right.

Examples

```r
x <- factor(
  c("Mon", "Tue", "Wed"),
  levels = c("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"),
  ordered = TRUE
)
x
fct_shift(x)
fct_shift(x, 2)
fct_shift(x, -1)
```
**fct_shuffle**

*Randomly permute factor levels*

**Description**

Randomly permute factor levels

**Usage**

fct_shuffle(f)

**Arguments**

- **f**
  A factor (or character vector).

**Examples**

```r
f <- factor(c("a", "b", "c"))
fct_shuffle(f)
fct_shuffle(f)
```

---

**fct_unify**

*Unify the levels in a list of factors*

**Description**

Unify the levels in a list of factors

**Usage**

fct_unify(fs, levels = lvls_union(fs))

**Arguments**

- **fs**
  A list of factors
- **levels**
  Set of levels to apply to every factor. Default to union of all factor levels

**Examples**

```r
fs <- list(factor("a"), factor("b"), factor(c("a", "b")))
fct_unify(fs)
```
**fct_unique**

Unique values of a factor, as a factor

**Description**

`fct_unique()` extracts the complete set of possible values from the levels of the factor, rather than looking at the actual values, like `unique()`.

`fct_unique()` only uses the values of `f` in one way: it looks for implicit missing values so that they can be included in the result.

**Usage**

```r
defunct_unique(f)
```

**Arguments**

- `f` A factor.

**Value**

A factor.

**Examples**

```r
f <- fct(letters[rpois(100, 10)])
unique(f)  # in order of appearance
fct_unique(f)  # in order of levels

f <- fct(letters[rpois(100, 2)], letters[1:20])
unique(f)  # levels that appear in data
fct_unique(f)  # all possible levels
```

---

**gss_cat**

A sample of categorical variables from the General Social survey

**Description**

A sample of categorical variables from the General Social survey

**Usage**

```r
gss_cat
```
Format

year  year of survey, 2000–2014 (every other year)
age  age. Maximum age truncated to 89.
marital  marital status
race  race
rincome  reported income
partyid  party affiliation
relig  religion
denom  denomination
tvhours  hours per day watching tv

Source
Downloaded from https://gssdataexplorer.norc.org/.

Examples

gss_cat

fct_count(gss_cat$relig)
fct_count(fct_lump(gss_cat$relig))

---

**lvls**  *Low-level functions for manipulating levels*

Description

`lvls_reorder` leaves values as they are, but changes the order. `lvls_revalue` changes the values of existing levels; there must be one new level for each old level. `lvls_expand` expands the set of levels; the new levels must include the old levels.

Usage

`lvls_reorder(f, idx, ordered = NA)`

`lvls_revalue(f, new_levels)`

`lvls_expand(f, new_levels)`

Arguments

`f`  A factor (or character vector).
`idx`  A integer index, with one integer for each existing level.
`ordered`  A logical which determines the "ordered" status of the output factor. NA preserves the existing status of the factor.
`new_levels`  A character vector of new levels.
Details

These functions are less helpful than the higher-level `fct_` functions, but are safer than the very low-level manipulation of levels directly, because they are more specific, and hence can more carefully check their arguments.

Examples

```r
f <- factor(c("a", "b", "c"))
lvls_reorder(f, 3:1)
lvls_revalue(f, c("apple", "banana", "carrot"))
lvls_expand(f, c("a", "b", "c", "d"))
```

---

**lvls_union**

*Find all levels in a list of factors*

**Description**

Find all levels in a list of factors

**Usage**

`lvls_union(fs)`

**Arguments**

`fs` A list of factors.

**Examples**

```r
fs <- list(factor("a"), factor("b"), factor(c("a", "b")))
lvls_union(fs)
```
Index

* datasets
  - gss_cat, 21
  - `%in%`, 12
  - as_factor, 2
  - factor(), 3
  - fct, 3
  - fct_anon, 4
  - fct_c, 5
  - fctCollapse, 6
  - fct_count, 6
  - fct_cross, 7
  - fct_drop, 8
  - fct_drop(), 9
  - fct_expand, 8
  - fct_expand(), 8
  - fct_infreq(fct_inorder), 9
  - fct_inorder, 9
  - fct_inseq(fct_inorder), 9
  - fct_lump, 10
  - fct_lump(), 14
  - fct_lump_lowfreq(fct_lump), 10
  - fct_lump_min(fct_lump), 10
  - fct_lump_n(fct_lump), 10
  - fct_lump_prop(fct_lump), 10
  - fct_match, 12
  - fct_na_level_to_value
    - (fct_na_value_to_level), 12
  - fct_na_value_to_level, 12
  - fct_other, 13
  - fct_other(), 11
  - fct_recode, 14
  - fct_relabel, 15
  - fct_relevel, 16
  - fct_reorder, 17
  - fct_reorder2(fct_reorder), 17
  - fct_rev, 18
  - fct_shift, 19
  - fct_shuffle, 20
  - fct_unify, 20
  - fct_unique, 21
  - first2(fct_reorder), 17
  - gss_cat, 21
  - is.na(), 12
  - last2(fct_reorder), 17
  - lvls, 22
  - lvlsExpand(lvls), 22
  - lvls_reorder(lvls), 22
  - lvls_revalue(lvls), 22
  - lvls_union, 23
  - rank(), 11
  - stats::relevel(), 16
  - unique(), 21

24