

Package ‘forcats’

January 23, 2017

Title Tools for Working with Categorical Variables (Factors)

Version 0.2.0

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 GPL-3

Encoding UTF-8

LazyData true

URL <http://forcats.tidyverse.org>, <https://github.com/tidyverse/forcats>

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

Depends R (>= 2.10)

Imports tibble, magrittr

RoxygenNote 5.0.1.9000

Suggests ggplot2, testthat, covr

NeedsCompilation no

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

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2017-01-23 16:39:48

R topics documented:

as_factor	2
fct_anon	3
fct_c	3
fct_collapse	4
fct_count	5
fct_drop	5

fct_expand	6
fct_explicit_na	6
fct_inorder	7
fct_lump	8
fct_other	9
fct_recode	10
fct_relabel	10
fct_relevel	11
fct_reorder	12
fct_rev	13
fct_shift	13
fct_shuffle	14
fct_unify	14
fct_unique	15
gss_cat	15
lvls	16
lvls_union	17

Index	18
--------------	-----------

as_factor	<i>Convert input to a factor.</i>
-----------	-----------------------------------

Description

Compared to base R, 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).

Usage

```
as_factor(x, ...)

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

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

Arguments

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

Details

This is a generic function.

Examples

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

```
gss_cat$relig %>% fct_count()
gss_cat$relig %>% fct_anon() %>% 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 useful way of patching together factors from multiple sources that really should have the same levels but don't.

Usage

```
fct_c(fs, ...)
```

Arguments

fs	A factor, or list of factors.
...	Individual factors

Examples

```
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 as the first argument
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, ...)
```

Arguments

f	A factor
...	A series of named character vectors. The levels in each vector will be replaced with the name.

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	<i>Count entries in a factor</i>
-----------	----------------------------------

Description

Count entries in a factor

Usage

```
fct_count(f, sort = FALSE)
```

Arguments

f	A factor
sort	If TRUE, sort the result so that most common values float to the top.

Value

A tibble with columns f and n

Examples

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

fct_drop	<i>Drop unused levels</i>
----------	---------------------------

Description

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

Usage

```
fct_drop(f, only)
```

Arguments

f	A factor
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.

Examples

```
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	<i>Add additional levels to a factor</i>
------------	--

Description

Add additional levels to a factor

Usage

```
fct_expand(f, ...)
```

Arguments

f	A factor.
...	Additional levels to add to the factor. Levels that already exist will be silently ignored.

Examples

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

fct_explicit_na	<i>Make missing values explicit</i>
-----------------	-------------------------------------

Description

This gives missing value an explicit factor level, ensuring that they appear in summaries and on plots.

Usage

```
fct_explicit_na(f, na_level = "(Missing)")
```

Arguments

f A factor.
na_level Level to use for missing values.

Examples

```
f1 <- factor(c("a", "a", NA, NA, "a", "b", NA, "c", "a", "c", "b"))  
table(f1)  
  
f2 <- fct_explicit_na(f1)  
table(f2)
```

fct_inorder	<i>Reorder factors levels by first appearance or frequency</i>
-------------	--

Description

Reorder factors levels by first appearance or frequency

Usage

```
fct_inorder(f, ordered = NA)  
  
fct_infreq(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.

Examples

```
f <- factor(c("b", "b", "a", "c", "c", "c"))  
f  
fct_inorder(f)  
fct_infreq(f)  
  
fct_inorder(f, ordered = TRUE)
```

fct_lump

*Lump together least/most common factor levels into "other"***Description**

Lump together least/most common factor levels into "other"

Usage

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

Arguments

f	A factor.
n, prop	If both n and prop are missing, fct_lump lumps together the least frequent levels into "other", while ensuring that "other" is still the smallest level. It's particularly useful in conjunction with fct_inorder() . Positive n preserves the most common n values. Negative n preserves the least common -n values. If there are ties, you will get at least abs(n) values. Positive prop, preserves values that appear at least prop of the time. Negative prop, preserves values that appear at most -prop of the time.
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

See Also

[fct_other\(\)](#) to convert specified levels to other.

Examples

```
x <- factor(rep(LETTERS[1:9], times = c(40, 10, 5, 27, 1, 1, 1, 1, 1)))
x %>% table()
x %>% fct_lump() %>% table()
x %>% fct_lump() %>% fct_inorder() %>% table()

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

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

# Use negative values to collapse the most common
fct_lump(x, n = -3)
```



```
fct_lump(x, prop = -0.1)

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

fct_other	<i>Replace levels with "other"</i>
-----------	------------------------------------

Description

Replace levels with "other"

Usage

```
fct_other(f, keep, drop, other_level = "Other")
```

Arguments

f	A factor.
keep, drop	keep will preserve listed levels, replacing all others other_level. drop will replace listed levels with other_level keep as they are.
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

```
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	<i>Change factor levels by hand</i>
------------	-------------------------------------

Description

Change factor levels by hand

Usage

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

Arguments

f	A factor.
...	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

```
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")
```

fct_relabel	<i>Automatically relabel factor levels, collapse as necessary</i>
-------------	---

Description

Automatically relabel factor levels, collapse as necessary

Usage

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

Arguments

f	A factor
fun	A function that is applied to each level. Must accept one character argument and return a character vector of the same length as its input.
...	Additional arguments to fun.

Examples

```
fct_count(gss_cat$rincome)

convert_income <- function(x) {
  regex <- "^(?:Lt |)[$]([0-9]+).*$"
  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
}

convert_income(levels(gss_cat$rincome))

rincome2 <- fct_relabel(gss_cat$rincome, convert_income)
fct_count(rincome2)
```

fct_relevel

Reorder factor levels by hand

Description

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

Usage

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

Arguments

f	A factor.
...	Character vector of levels. Any levels not mentioned will be left in existing order, after the explicitly mentioned levels.
after	Where should the new values be placed?

Examples

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

# 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.

Usage

```
fct_reorder(f, x, fun = median, ..., .desc = FALSE)
```

```
fct_reorder2(f, x, y, fun = last2, ..., .desc = TRUE)
```

Arguments

f	A factor
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.
fun	An summary function. It should take one vector for fct_reorder, and two vectors for fct_reorder2.
...	Other arguments passed on to fun. A common argument is na.rm = TRUE.
.desc	Order in descending order? Note the default is different between fct_reorder and fct_reorder2, in order to match the default ordering of factors in the legend.

Examples

```
boxplot(Sepal.Width ~ Species, data = iris)
boxplot(Sepal.Width ~ fct_reorder(Species, Sepal.Width), data = iris)
boxplot(Sepal.Width ~ fct_reorder(Species, Sepal.Width, .desc = TRUE), data = iris)

chks <- subset(ChickWeight, as.integer(Chick) < 10)
chks <- transform(chks, Chick = fct_shuffle(Chick))
```

```
if (require("ggplot2")) {
  ggplot(chks, aes(Time, weight, colour = Chick)) +
    geom_point() +
    geom_line()

  # Note that lines match order in legend
  ggplot(chks, aes(Time, weight, colour = fct_reorder2(Chick, Time, weight))) +
    geom_point() +
    geom_line() +
    labs(colour = "Chick")
}
```

fct_rev*Reverse order of factor levels*

Description

This is sometimes useful when plotting a factor

Usage

```
fct_rev(f)
```

Arguments

f A factor

Examples

```
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

```
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	<i>Randomly permute factor levels</i>
-------------	---------------------------------------

Description

Randomly permute factor levels

Usage

```
fct_shuffle(f)
```

Arguments

f A factor

Examples

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

fct_unify	<i>Unify the levels in a list of factors</i>
-----------	--

Description

Unify the levels in a list of factors

Usage

```
fct_unify(fs, levels = lvl_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

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

fct_unique	<i>Unique values of a factor</i>
------------	----------------------------------

Description

Unique values of a factor

Usage

```
fct_unique(f)
```

Arguments

f A factor

Examples

```
f <- factor(letters[rpois(100, 10)])

unique(f)        # in order of appearance
fct_unique(f) # in order of levels
```

gss_cat	<i>A sample of categorical variables from the General Social survey</i>
---------	---

Description

A sample of categorical variables from the General Social survey

Usage

```
gss_cat
```

Format

year year of survey, 2000–2014
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 is, but changes the order. `lvls_revalue` changes the values of existing levels; there must be one new level for each old level. `lvls_extend` extends the set of levels; the new level must include the old levels.

Usage

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

lvls_revalue(f, new_levels)

lvls_expand(f, new_levels)
```

Arguments

<code>f</code>	A factor
<code>idx</code>	A integer index, with one integer for each existing level
<code>ordered</code>	A logical which determines the "ordered" status of the output factor. NA preserves the existing status of the factor.
<code>new_levels</code>	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

```
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

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

Index

*Topic **datasets**

gss_cat, [15](#)

as_factor, [2](#)

fct_anon, [3](#)

fct_c, [3](#)

fct_collapse, [4](#)

fct_count, [5](#)

fct_drop, [5](#)

fct_expand, [6](#)

fct_explicit_na, [6](#)

fct_infreq(fct_inorder), [7](#)

fct_inorder, [7, 8](#)

fct_lump, [8](#)

fct_lump(), [9](#)

fct_other, [9](#)

fct_other(), [8](#)

fct_recode, [10](#)

fct_relabel, [10](#)

fct_relevel, [11](#)

fct_reorder, [12](#)

fct_reorder2(fct_reorder), [12](#)

fct_rev, [13](#)

fct_shift, [13](#)

fct_shuffle, [14](#)

fct_unify, [14](#)

fct_unique, [15](#)

gss_cat, [15](#)

lvls, [16](#)

lvls_expand(lvls), [16](#)

lvls_reorder(lvls), [16](#)

lvls_revalue(lvls), [16](#)

lvls_union, [17](#)

rank(), [8](#)

stats::relevel(), [11](#)