Package 'forcats'

January 27, 2021

Title Tools for Working with Categorical Variables (Factors)

Version 0.5.1

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

URL https://forcats.tidyverse.org,
https://github.com/tidyverse/forcats

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

Depends R (>= 3.2)

Imports ellipsis, magrittr, rlang, tibble

Suggests covr, dplyr, ggplot2, knitr, readr, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

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

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2021-01-27 19:50:02 UTC

R topics documented:

---

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

```
as_factor(x, ...)

## S3 method for class 'factor'
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

```
# 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_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("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

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

Arguments

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

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 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)
```

Arguments

`f`	A factor (or character vector).
`sort`	If TRUE, sort the result so that the most common values float to the top.
`prop`	If TRUE, compute the fraction of marginal table.

Value

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

Examples

```
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

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

Arguments

...	<dynamic-dots> Additional factors or character vectors.
sep	A character string to separate the levels
keep_empty	If TRUE, keep combinations with no observations as levels

Value

The new factor

Examples

```
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

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

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

```
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

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

Arguments

f	A factor (or character vector).
...	Additional levels to add to the factor. Levels that already exist will be silently ignored.

See Also

[fct_drop()](#) to drop unused factor levels.

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   *Make missing values explicit*

Description

This gives missing values 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 (or character vector).

na_level   Level to use for missing values: this is what NAs will be changed to.

Examples

```
f1 <- factor(c("a", "a", NA, NA, "a", "b", NA, "c", "a", "c", "b"))
fct_count(f1)

f2 <- fct_explicit_na(f1)
fct_count(f2)
```

---

fct_inorder   *Reorder factor levels by first appearance, frequency, or numeric order*

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

```
fct_inorder(f, ordered = NA)

fct_infreq(f, 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.

Examples

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

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

`fct_lump`	*Lump together factor levels into "other"*

Description

A family for lumping together levels that meet some criteria.

- `fct_lump_min()`: lumps levels that appear fewer than `min` times.
- `fct_lump_prop()`: lumps levels that appear in fewer `prop * n` times.
- `fct_lump_n()` lumps all levels except for the `n` most frequent (or least frequent if `n < 0`)
- `fct_lump_lowfreq()` 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

```
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, 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.
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

```
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
table(fct_lump_min(x, min = 10))
table(fct_lump_min(x, min = 15))
```

---

fct_match   *Test for presence of levels in a factor*

Description

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

Usage

```
fct_match(f, lvls)
```

Arguments

f	A factor (or character vector).
lvls	A character vector specifying levels to look for.

Value

A logical vector

Examples

```
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)
```

---

fct_other   *Replace levels with "other"*

Description

Replace levels with "other"

Usage

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

```
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

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

Arguments

.f	A factor (or character vector).
...	<[dynamic-dots](#)> 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")

# 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   *Automatically relabel factor levels, collapse as necessary*

Description

Automatically relabel factor levels, collapse 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 <- "^(?: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 $", num_income)
  x
}
fct_count(gss_cat$rincome)
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 (or character vector).
...	Either a function (or formula), or character levels.
	A function will be called with the current levels, and the return value (which must be a character vector) will be used to relevel the function.
	Any levels not mentioned will be left in their existing order, after the explicitly mentioned levels. Supports tidy dots.
after	Where should the new values be placed?

Examples

```
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

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

fct_reorder2(.f, .x, .y, .fun = last2, ..., .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.
.fun	n summary function. It should take one vector for `fct_reorder`, and two vectors for `fct_reorder2`, and return a single value.
...	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

```
df <- tibble::tribble(
  ~color,     ~a, ~b,
  "blue",      1,  2,
  "green",     6,  2,
  "purple",    3,  3,
  "red",       2,  3,
  "yellow",    5,  1
)
df$color <- factor(df$color)
fct_reorder(df$color, df$a, min)
fct_reorder2(df$color, df$a, df$b)

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 (or character vector).

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	*Randomly permute factor levels*

Description

Randomly permute factor levels

Usage

```
fct_shuffle(f)
```

Arguments

f	A factor (or character vector).

Examples

```
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

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

---

fct_unique   *Unique values of a factor*

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`	*A sample of categorical variables from the General Social survey*

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 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

```
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