Factors

David Gerard

2019-02-26

Learning Objectives

- Manipulating factors.
- Chapter 15 of RDS.
- Wrangling Categorical Data in R.
- 8.2: Chimeras of the R Inferno
- Factors with forcats Cheat Sheet.

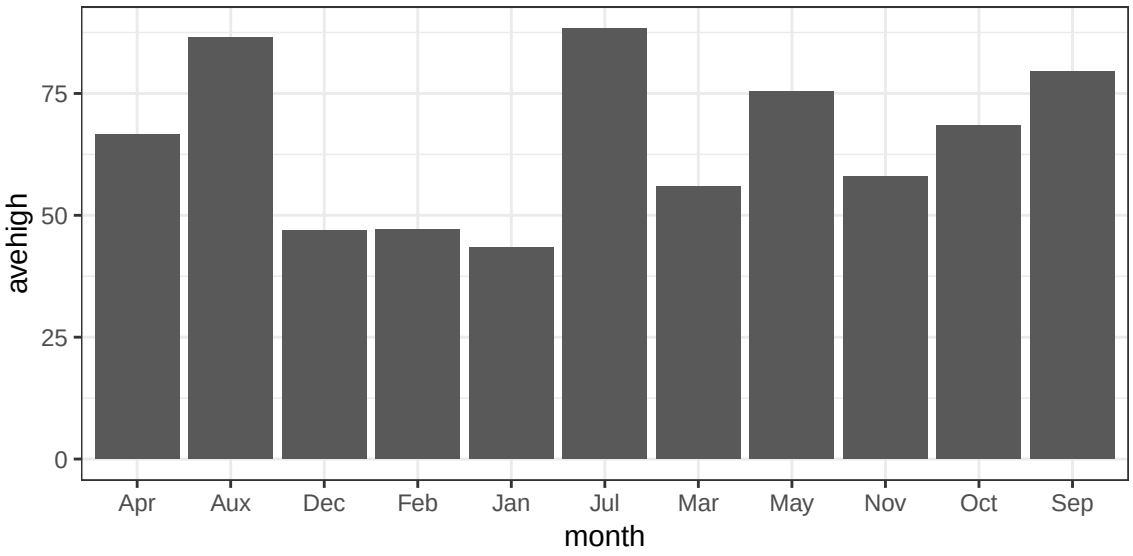
Factors

- A "factor" is R's way to say that a variable is categorical (places observational/experimental units into different groups or categories based on its values.).
- A factor is different from a character in that:
    1. There is a small predefined set of "levels" (possible values) of a factor, but not of a character.
    2. There is an ordering for the levels of a factor
        - Useful when determining the order to plot something.
        - Useful when doing ordered logistic regression.
- Consider the following data frame for average highs in DC for each month.

```
library(tidyverse)
dcclimate <- tribble(~month, ~avehigh,
                     ##----/---------
                     "Jan",  43.4,
                     "Feb",  47.1,
                     "Mar",  55.9,
                     "Apr",  66.6,
                     "May",  75.4,
                     "Jul",  88.4,
                     "Aux",  86.5,
                     "Sep",  79.5,
                     "Oct",  68.4,
                     "Nov",  57.9,
                     "Dec",  46.8)
```

- The weather for June is missing and the 3-letter abbreviation for August is incorrect. We would like to notice both of these.
- Also, when we plot the data, we would prefer the order to be the same as that for the order of the months of the year.

```
ggplot(dcclimate, aes(x = month, y = avehigh)) +
  geom_col()
```



- Factors help us with all of these issues.
- You have to be **very** careful about factors.

```
x  <- c("51", "32", "15", "2", "32")
xf <- factor(x)
as.numeric(x)
```

```
## [1] 51 32 15  2 32
```

```
as.numeric(xf)
```

```
## [1] 4 3 1 2 3
```

```
as.numeric("Hello")
```

```
## Warning: NAs introduced by coercion
```

```
## [1] NA
```

```
as.numeric(factor("Hello"))
```

```
## [1] 1
```

```
fac1 <- factor(c("x1", "x2", "x3"))
fac2 <- factor(c("y1", "y2", "y3"))
c(fac1, fac2)
```

```
## [1] 1 2 3 1 2 3
```

Creating Factors

- Use `factor()` or `parse_factor()` to create a factor variable
- `parse_factor()` returns better warnings, so I would recommend always using that.

```
monthvec <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
dcclimate %>%
  mutate(monthfc = factor(month, levels = monthvec)) ->
  dcclimate

dcclimate %>%
  mutate(monthfc2 = parse_factor(month, levels = monthvec)) ->
  dcclimate
```

```
## Warning: 1 parsing failure.
## row col           expected actual
##   7  -- value in level set    Aux
```

```
dcclimate$monthfc
```

```
##  [1] Jan  Feb  Mar  Apr  May  Jul  <NA> Sep  Oct  Nov  Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

- If you do not specify the `levels` argument, R will assume that the levels are the unique values of the vector.
  - `factor()` takes the order of the levels to be the same order returned by `sort()`.
  - `parse_factor()` takes the order of the levels to be the same order as the order of the value introduced.

```
x <- c("A", "string", "vector", "is", "a", "string", "vector")
factor(x)
```

```
## [1] A      string vector is     a      string vector
## Levels: a A is string vector
```

```
sort(unique(x))
```

```
## [1] "a"      "A"      "is"     "string" "vector"
```

```
parse_factor(x)
```

```
## [1] A      string vector is     a      string vector
## Levels: A string vector is a
```

- You can always see the levels of a factor (and their order) using the `levels()` function

```
levels(dcclimate$monthfc)
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
## [12] "Dec"
```

- Other options are the `fct_unique()` and `fct_count()` functions from the forcats package.

```
fct_unique(dcclimate$monthfc)
```

```
##  [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
fct_count(dcclimate$monthfc)
```

```
## # A tibble: 13 x 2
##    f         n
##    <fct> <int>
##  1 Jan       1
##  2 Feb       1
##  3 Mar       1
##  4 Apr       1
##  5 May       1
##  6 Jun       0
##  7 Jul       1
##  8 Aug       0
##  9 Sep       1
## 10 Oct       1
## 11 Nov       1
## 12 Dec       1
## 13 <NA>      1
```

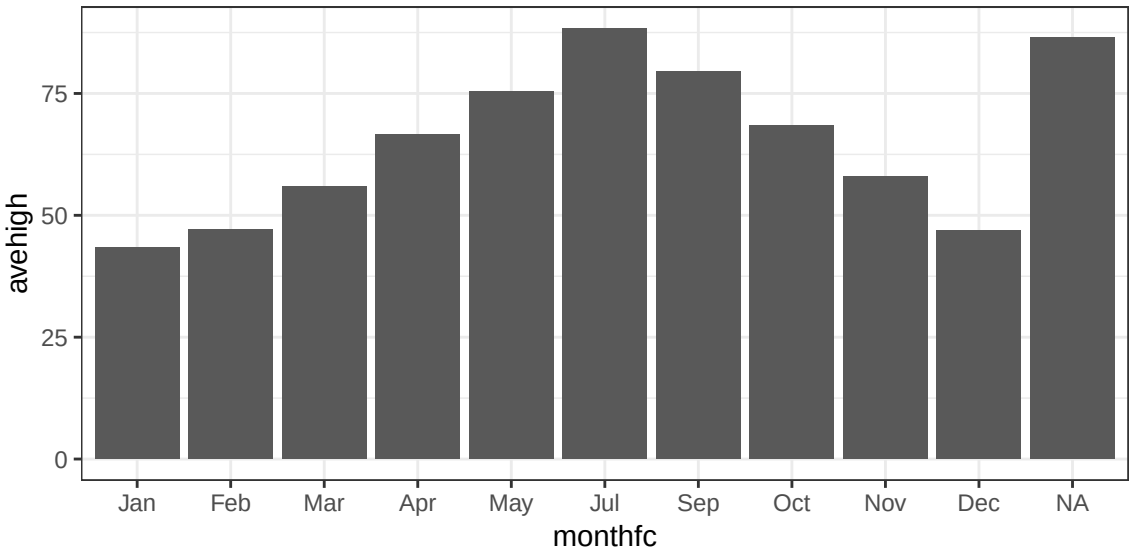
- You can count the number of levels with `nlevels()`.

```
nlevels(dcclimate$monthfc)
```

```
## [1] 12
```

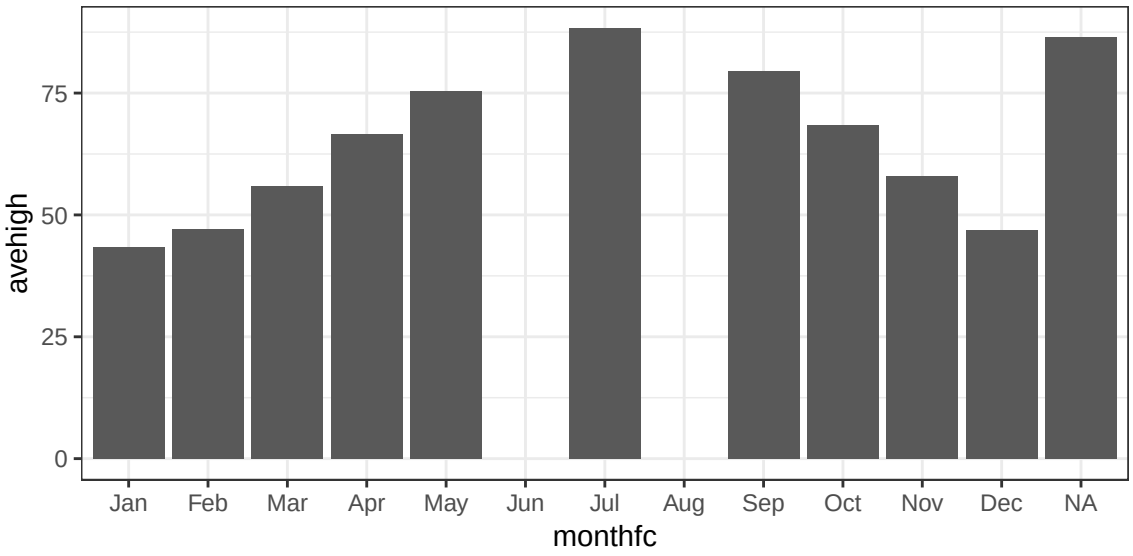
- Once we have a factor variable, the order of the aesthetic map is set in ggplot.

```
ggplot(dcclimate, aes(x = monthfc, y = avehigh)) +
  geom_col()
```



- We can include missing levels by using the `drop = FALSE` argument in the appropriate scale call:

```
ggplot(dcclimate, aes(x = monthfc, y = avehigh)) +
  geom_col() +
  scale_x_discrete(drop = FALSE)
```



forcats

- forcats is an R package which makes two things much easier in R:
  - Changing the order of the levels of the factor variable.
  - Changing the levels of the factor variable.
- It also a few other helper functions for factors.

- All forcat functions begin with `fct_`. So you can type "fct_" then use tab-completion to scroll through the possible functions.
- forcats is a part of the tidyverse, so you don't need to load it separately when you load the tidyverse.

Changing the Order of the Levels

- Consider the subset of the General Social Survey stored in the `gss_cat` data in forcats.

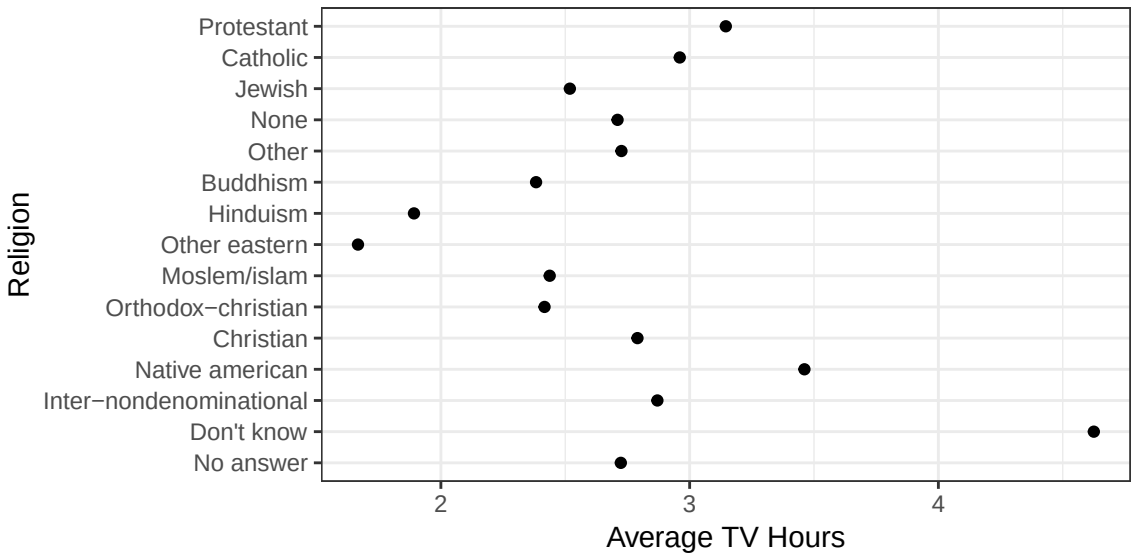
```
data(gss_cat)
glimpse(gss_cat)
```

```
## Observations: 21,483
## Variables: 9
## $ year    <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, ...
## $ marital <fct> Never married, Divorced, Widowed, Never married, Divor...
## $ age     <int> 26, 48, 67, 39, 25, 25, 36, 44, 44, 47, 53, 52, 52, 51...
## $ race    <fct> White, White, White, White, White, White, White, White...
## $ rincome <fct> $8000 to 9999, $8000 to 9999, Not applicable, Not appl...
## $ partyid <fct> "Ind,near rep", "Not str republican", "Independent", "...
## $ relig   <fct> Protestant, Protestant, Protestant, Orthodox-christian...
## $ denom   <fct> Southern baptist, Baptist-dk which, No denomination, N...
## $ tvhours <int> 12, NA, 2, 4, 1, NA, 3, NA, 0, 3, 2, NA, 1, NA, 1, 7, ...
```

- You often want to change the order of the levels of a factor to make plots more insightful.

```
gss_cat %>%
  group_by(relig) %>%
  summarize(tvhours_mean = mean(tvhours, na.rm = TRUE)) ->
  tvdat

ggplot(tvdat, aes(x = tvhours_mean, y = relig)) +
  geom_point() +
  xlab("Average TV Hours") +
  ylab("Religion")
```



- `fct_reorder()` reorders the levels of a factor according to some values of another variable. The arguments are:
  - `f`: The factor vector.
  - `x`: A numeric vector used to reorder the levels.
  - `fun`: A function applied to `x`, the result of which will be used to order the levels of `f`.

```
levels(tvdat$relig)
```

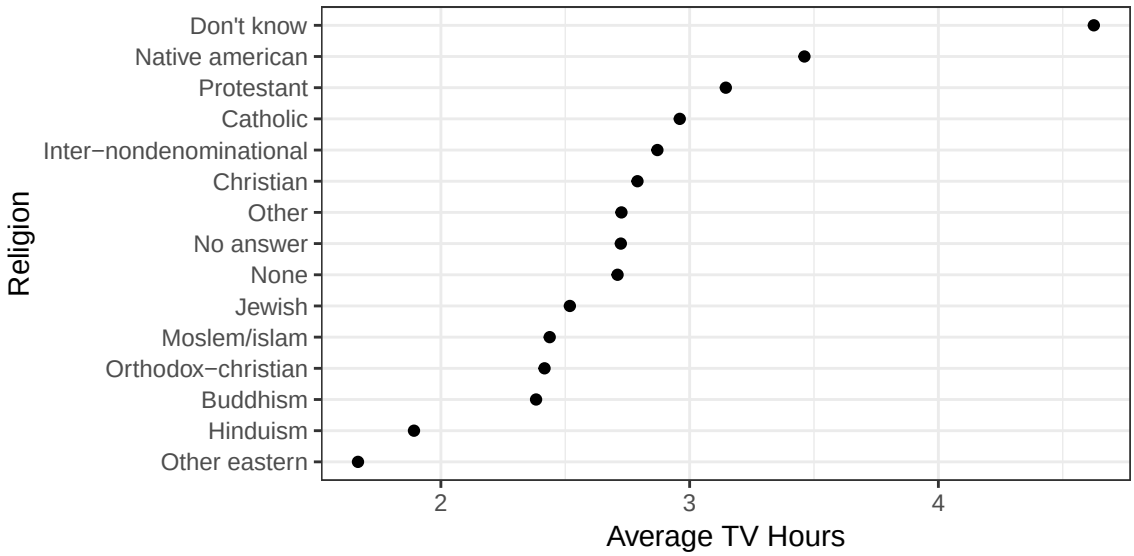
```
##  [1] "No answer"               "Don't know"
##  [3] "Inter-nondenominational" "Native american"
##  [5] "Christian"               "Orthodox-christian"
##  [7] "Moslem/islam"            "Other eastern"
##  [9] "Hinduism"                "Buddhism"
## [11] "Other"                   "None"
## [13] "Jewish"                  "Catholic"
## [15] "Protestant"              "Not applicable"
```

```
tvdat %>%
  mutate(relig = fct_reorder(relig, tvhours_mean)) ->
  tvdat
levels(tvdat$relig)
```

```
##  [1] "Other eastern"           "Hinduism"
##  [3] "Buddhism"                "Orthodox-christian"
##  [5] "Moslem/islam"            "Jewish"
##  [7] "None"                    "No answer"
##  [9] "Other"                   "Christian"
## [11] "Inter-nondenominational" "Catholic"
## [13] "Protestant"              "Native american"
## [15] "Don't know"              "Not applicable"
```

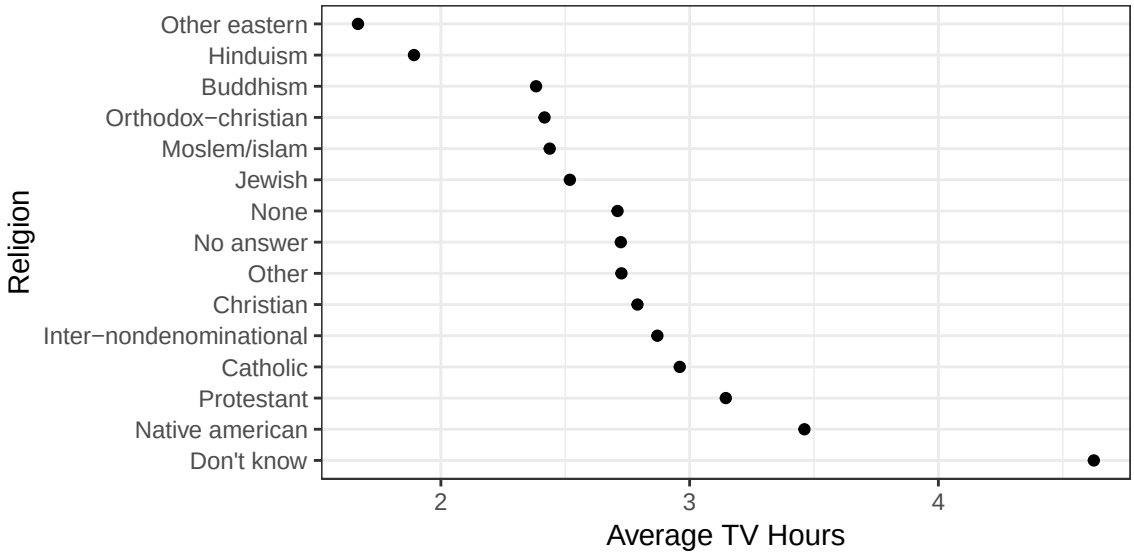
- The plot now reorders the y-axis according to the new level order.

```
ggplot(tvdat, aes(x = tvhours_mean, y = relig)) +
  geom_point() +
  xlab("Average TV Hours") +
  ylab("Religion")
```



- `fct_rev()` reverses the order of the factors.

```
tvdat %>%
  mutate(relig = fct_rev(relig)) %>%
  ggplot(aes(x = tvhours_mean, y = relig)) +
    geom_point() +
    xlab("Average TV Hours") +
    ylab("Religion")
```



- `fct_relevel()` allows you to move existing levels to any location.

```
## Moves "None" to first level
fct_relevel(tvdat$relig, "None") %>%
  levels()
```

```
##  [1] "None"                    "Other eastern"
##  [3] "Hinduism"                "Buddhism"
##  [5] "Orthodox-christian"      "Moslem/islam"
##  [7] "Jewish"                  "No answer"
##  [9] "Other"                   "Christian"
## [11] "Inter-nondenominational" "Catholic"
## [13] "Protestant"              "Native american"
## [15] "Don't know"              "Not applicable"
```

```
## Moves "None" to the third level
fct_relevel(tvdat$relig, "None", after = 2L) %>%
  levels()
```

```
##  [1] "Other eastern"           "Hinduism"
##  [3] "None"                    "Buddhism"
##  [5] "Orthodox-christian"      "Moslem/islam"
##  [7] "Jewish"                  "No answer"
##  [9] "Other"                   "Christian"
## [11] "Inter-nondenominational" "Catholic"
## [13] "Protestant"              "Native american"
## [15] "Don't know"              "Not applicable"
```

```
## Moves "None" to the last level
fct_relevel(tvdat$relig, "None", after = nlevels(tvdat$relig)) %>%
  levels()
```

```
##  [1] "Other eastern"           "Hinduism"
##  [3] "Buddhism"                "Orthodox-christian"
##  [5] "Moslem/islam"            "Jewish"
##  [7] "No answer"               "Other"
##  [9] "Christian"               "Inter-nondenominational"
## [11] "Catholic"                "Protestant"
## [13] "Native american"         "Don't know"
## [15] "Not applicable"          "None"
```

```
## Returns a warning because "Cthulhuism" is not a level
fct_relevel(tvdat$relig, "Cthulhuism")
```

```
## Warning: Unknown levels in `f`: Cthulhuism
```

```
##  [1] No answer               Don't know
##  [3] Inter-nondenominational Native american
##  [5] Christian               Orthodox-christian
##  [7] Moslem/islam            Other eastern
##  [9] Hinduism                Buddhism
## [11] Other                   None
## [13] Jewish                  Catholic
## [15] Protestant
## 16 Levels: Other eastern Hinduism Buddhism ... Not applicable
```

- **Exercise**: Reorder the levels of the `partyid` variable so that the levels are in alphabetical order.
- **Exercise**: Move the `"Not applicable"` level to the front in the `rincome` variable.

Modify Factor Levels

- Let's look at the levels of `partyid` in `gss_cat`.

```
levels(gss_cat$partyid)
```

```
##  [1] "No answer"          "Don't know"         "Other party"
##  [4] "Strong republican"  "Not str republican" "Ind,near rep"
##  [7] "Independent"        "Ind,near dem"       "Not str democrat"
## [10] "Strong democrat"
```

- Use `fct_recode()` to change the levels.

```
gss_cat %>%
  mutate(partyid = fct_recode(partyid,
                              "Republican, strong"    = "Strong republican",
                              "Republican, weak"      = "Not str republican",
                              "Independent, near rep" = "Ind,near rep",
                              "Independent, near dem" = "Ind,near dem",
                              "Democrat, weak"        = "Not str democrat",
                              "Democrat, strong"      = "Strong democrat"
                              )) ->
  gss_cat
levels(gss_cat$partyid)
```

```
##  [1] "No answer"             "Don't know"
##  [3] "Other party"           "Republican, strong"
##  [5] "Republican, weak"      "Independent, near rep"
##  [7] "Independent"           "Independent, near dem"
##  [9] "Democrat, weak"        "Democrat, strong"
```

- New level goes on the left of the equals sign. Old level goes on the right.
- **Exercise**: Modify the factor levels of `marital` to be abbreviations of their long-names. For example, "Divorced" can just be "D"

Other Useful Functions.

- `fct_c()`: is the safe way to combine factor vectors.

```
fc1 <- parse_factor(c("A", "B"))
fc1
```

```
## [1] A B
## Levels: A B
```

```
fc2 <- parse_factor(c("C", "D"))
fc2
```

```
## [1] C D
## Levels: C D
```

```
fct_c(fc1, fc2)
```

```
## [1] A B C D
## Levels: A B C D
```

- `fct_collapse()`: combine multiple levels into one level.

```
fc <- parse_factor(c("A", "B", "C", "A", "B", "C"))
fc
```

```
## [1] A B C A B C
## Levels: A B C
```

```
fct_collapse(fc, "blah" = c("A", "B"))
```

```
## [1] blah blah C    blah blah C
## Levels: blah C
```

- `fct_drop()`: removes any levels that are unused.

```
fc <- parse_factor(c("A", "B"), levels = c("A", "B", "C"))
fc
```

```
## [1] A B
## Levels: A B C
```

```
fct_drop(fc)
```

```
## [1] A B
## Levels: A B
```

- `fct_expand()`: adds a new level.

```
fc <- parse_factor(c("A", "B"))
fc
```

```
## [1] A B
## Levels: A B
```

```
fct_expand(fc, "C")
```

```
## [1] A B
## Levels: A B C
```

- `fct_infreq()`: Order by frequency of a level.

```
fc <- parse_factor(c("A", "B", "C", "B", "C", "C"))
fct_count(fc)
```

```
## # A tibble: 3 x 2
##   f         n
##   <fct> <int>
## 1 A         1
## 2 B         2
## 3 C         3
```

```
fct_infreq(fc) %>%
  fct_count()
```

```
## # A tibble: 3 x 2
##   f         n
##   <fct> <int>
## 1 C         3
## 2 B         2
## 3 A         1
```