Factors

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.
- Forcats Overview.

Factors

- A "factor" is R's way to say that a variable is categorical (puts observational/experimental units into different groups or categories based on their 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,
```

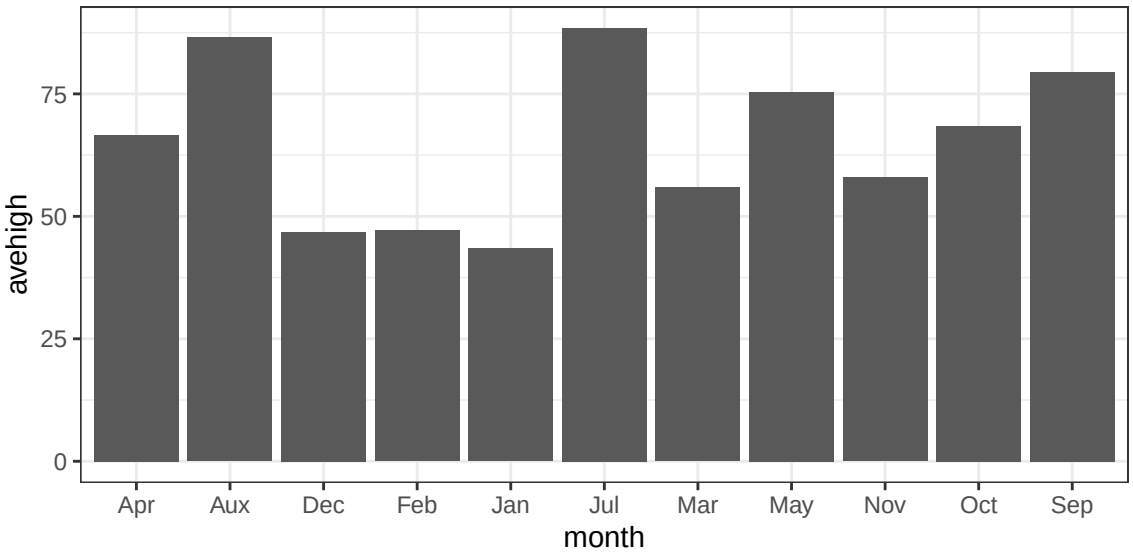
```
                          "May",  75.4,
                          "Jul",  88.4,
                          "Aux",  86.5,
                          "Sep",  79.5,
                          "Oct",  68.4,
                          "Nov",  57.9,
                          "Dec",  46.8)
```

```
dcclimate
```

```
# A tibble: 11 x 2
   month avehigh
   <chr>   <dbl>
 1 Jan      43.4
 2 Feb      47.1
 3 Mar      55.9
 4 Apr      66.6
 5 May      75.4
 6 Jul      88.4
 7 Aux      86.5
 8 Sep      79.5
 9 Oct      68.4
10 Nov      57.9
11 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")
x
```

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

```
class(x)
```

```
[1] "character"
```

```
as.numeric(x)
```

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

- Order: By default, R orders the levels of a factor in alphabetical order. In this case, since the levels are character strings, they are ordered alphabetically. So, the levels in **`xf`** would be "15", "2", "32", and "51".
- "15" comes before "2" because in alphabetical order, "1" comes before "2".
- "2" comes before "32" because in alphabetical order, "2" comes before "3".

- “32” comes before “51” because in alphabetical order, “3” comes before “5”.
- “51” comes last because it starts with the highest numerical digit among the strings.

```
    xf <- factor(x)
    xf
```

```
[1] 51 32 15 2  32
Levels: 15 2 32 51
```

- When you convert a factor to numeric in R using **`as.numeric()`**, it doesn’t return the original values of the levels of the factor, but rather the underlying integer codes that represent each level.
- In our case:

1. The first level in our factor **`xf`** is “15”, which gets assigned the numeric code 1.
2. The second level is “2”, which gets assigned the numeric code 2.
3. The third level is “32”, which gets assigned the numeric code 3.
4. The fourth level is “51”, which gets assigned the numeric code 4.

So, when you use **`as.numeric(xf)`**, it returns the numeric codes corresponding to each level of the factor, resulting in the output **`4 3 1 2 3`**.

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

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

```
factor("Hello")
```

```
[1] Hello
Levels: Hello
```

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

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

```
[1] NA
```

```r
  factor("Hello")
```

```
[1] Hello
Levels: Hello
```

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

```
[1] 1
```

```r
      fac1 <- factor(c("x1", "x2", "x3"))
  fac1
```

```
[1] x1 x2 x3
Levels: x1 x2 x3
```

```r
      fac2 <- factor(c("y1", "y2", "y3"))
  fac2
```

```
[1] y1 y2 y3
Levels: y1 y2 y3
```

```r
      c(fac1, fac2)
```

```
[1] x1 x2 x3 y1 y2 y3
Levels: x1 x2 x3 y1 y2 y3
```

```r
  xf
```

```
[1] 51 32 15 2  32
Levels: 15 2 32 51
```

```r
  levels(xf)
```

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

- If you are 100% sure that all levels are numeric and are incorrectly specified as factors, then do the following to convert to numeric:

```
    parse_number(levels(xf))
```

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

Creating Factors

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

```
dcclimate
```

```
# A tibble: 11 x 2
   month avehigh
   <chr>   <dbl>
 1 Jan      43.4
 2 Feb      47.1
 3 Mar      55.9
 4 Apr      66.6
 5 May      75.4
 6 Jul      88.4
 7 Aux      86.5
 8 Sep      79.5
 9 Oct      68.4
10 Nov      57.9
11 Dec      46.8
```

```
    monthvec <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
```

```
    dcclimate %>%
      mutate(monthfc = factor(month, levels = monthvec)) ->
      dcclimate

    dcclimate
```

```
# A tibble: 11 x 3
   month avehigh monthfc
```

```
   <chr>   <dbl> <fct>
 1 Jan      43.4 Jan
 2 Feb      47.1 Feb
 3 Mar      55.9 Mar
 4 Apr      66.6 Apr
 5 May      75.4 May
 6 Jul      88.4 Jul
 7 Aux      86.5 <NA>
 8 Sep      79.5 Sep
 9 Oct      68.4 Oct
10 Nov      57.9 Nov
11 Dec      46.8 Dec
```

```
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
```

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

```
Warning: There was 1 warning in `mutate()`.
i In argument: `monthfc2 = parse_factor(month, levels = monthvec)`.
Caused by warning:
! 1 parsing failure.
row col           expected actual
  7  -- value in level set    Aux
```

```
    dcclimate
```

```
# A tibble: 11 x 4
   month avehigh monthfc monthfc2
   <chr>   <dbl> <fct>   <fct>
 1 Jan      43.4 Jan     Jan
 2 Feb      47.1 Feb     Feb
 3 Mar      55.9 Mar     Mar
 4 Apr      66.6 Apr     Apr
 5 May      75.4 May     May
```

```
 6 Jul      88.4 Jul     Jul
 7 Aux      86.5 <NA>    <NA>
 8 Sep      79.5 Sep     Sep
 9 Oct      68.4 Oct     Oct
10 Nov      57.9 Nov     Nov
11 Dec      46.8 Dec     Dec
```

```
dcclimate$monthfc2
```

```
 [1] Jan  Feb  Mar  Apr  May  Jul  <NA> Sep  Oct  Nov  Dec
attr(,"problems")
# A tibble: 1 x 4
    row   col expected           actual
  <int> <int> <chr>              <chr>
1     7    NA value in level set Aux
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" "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  <NA>
Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
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
```

```
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
```

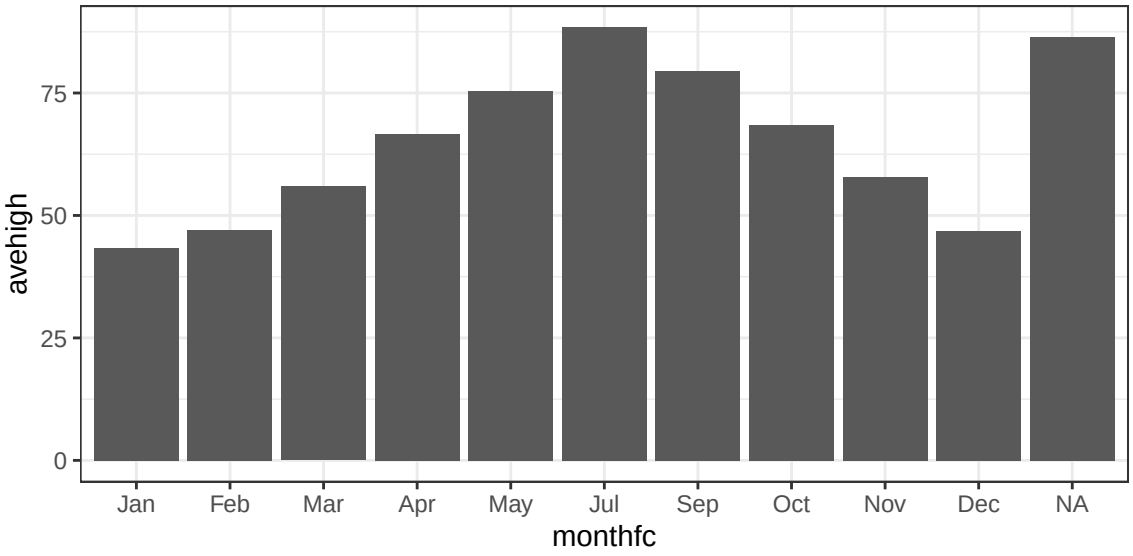
- You can count the number of levels with `nlevels()`.

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

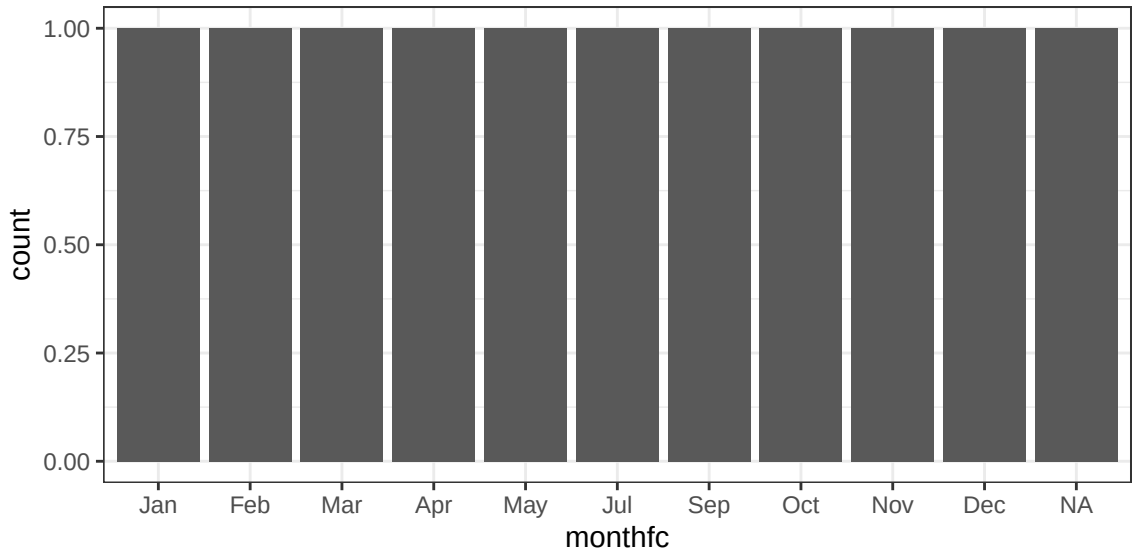
```
[1] 12
```

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

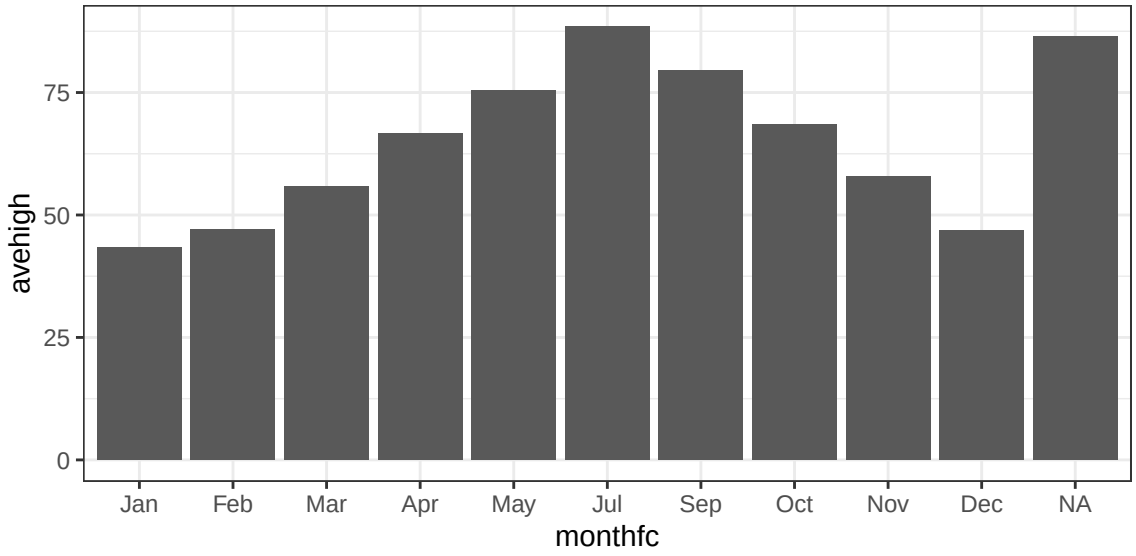
```
ggplot(dcclimate, aes(x = monthfc, y = avehigh)) +
  geom_col()   # if you want to specify the height use geom_col()
```




```
ggplot(dcclimate, aes(x = monthfc)) +
  geom_bar()    # it uses frequency by default
```



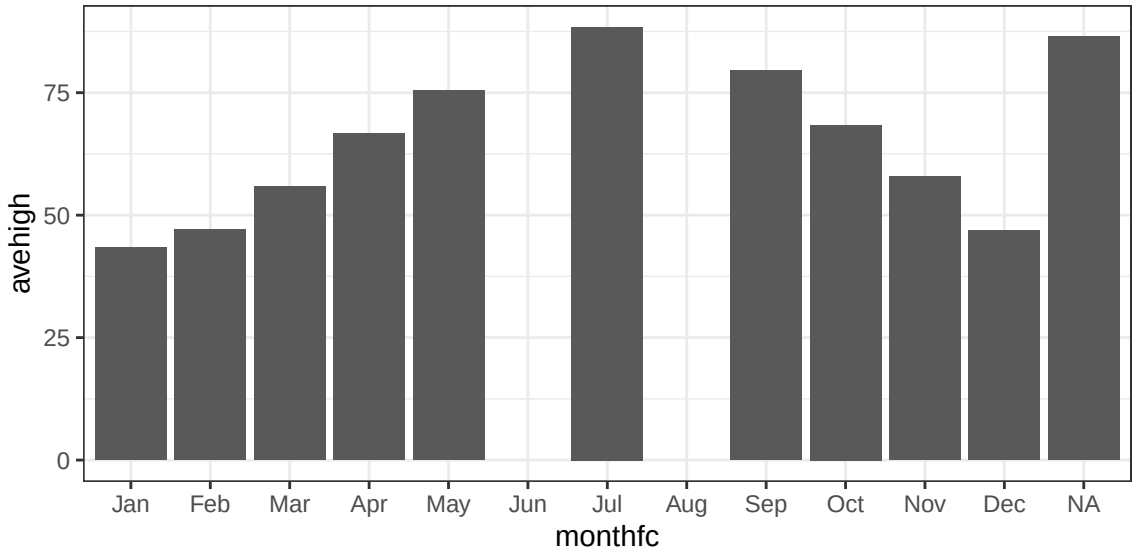

```
ggplot(dcclimate, aes(x = monthfc, y = avehigh)) +
  geom_bar(stat = "identity") # by default it use count. If you want to use the value
```




```
# data use stat="identity"
```

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

```r
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 {forcats} functions begin with `fct_`. So you can type "fct_" then use tab-completion to scroll through the possible functions. The goal of the {forcats} is to provide a suite of tools that solves common problem with factors.
- 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)
```

```
Rows: 21,483
Columns: 9
$ year    <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 20~
$ marital <fct> Never married, Divorced, Widowed, Never married, Divorced, Mar~
$ age     <int> 26, 48, 67, 39, 25, 25, 36, 44, 44, 47, 53, 52, 52, 51, 52, 40~
$ race    <fct> White, White, White, White, White, White, White, White, White,~
$ rincome <fct> $8000 to 9999, $8000 to 9999, Not applicable, Not applicable, ~
$ partyid <fct> "Ind,near rep", "Not str republican", "Independent", "Ind,near~
$ relig   <fct> Protestant, Protestant, Protestant, Orthodox-christian, None, ~
$ denom   <fct> "Southern baptist", "Baptist-dk which", "No denomination", "No~
$ tvhours <int> 12, NA, 2, 4, 1, NA, 3, NA, 0, 3, 2, NA, 1, NA, 1, 7, NA, 3, 3~
```

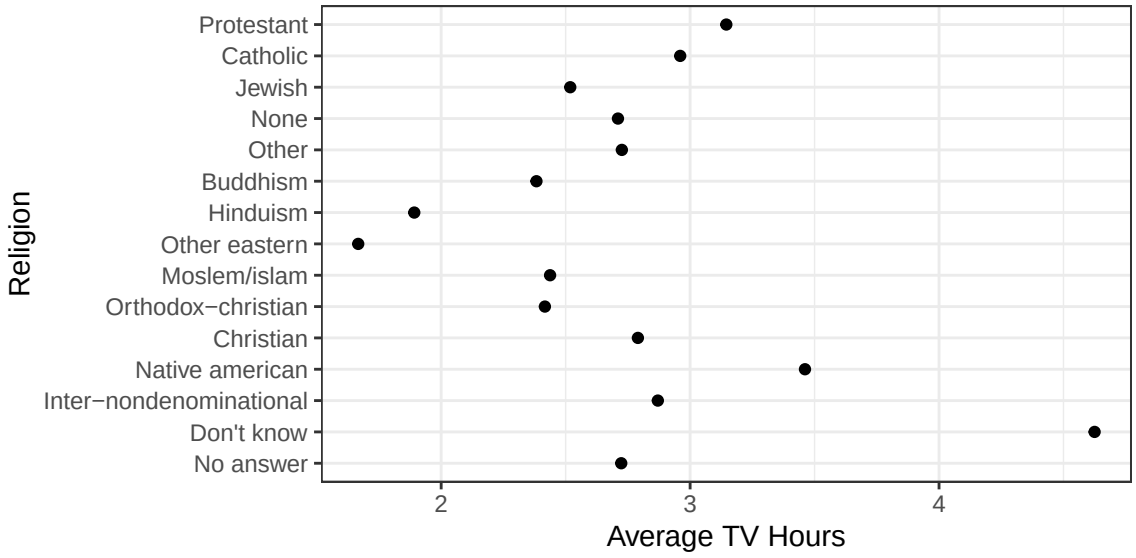
- You often want to change the order of the levels of a factor to make plots more insightful.

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

tvdat
```

```
# A tibble: 15 x 2
   relig                  tvhours_mean
   <fct>                         <dbl>
 1 No answer                      2.72
 2 Don't know                     4.62
 3 Inter-nondenominational        2.87
 4 Native american                3.46
 5 Christian                      2.79
 6 Orthodox-christian             2.42
 7 Moslem/islam                   2.44
 8 Other eastern                  1.67
 9 Hinduism                       1.89
10 Buddhism                       2.38
11 Other                          2.73
12 None                           2.71
13 Jewish                         2.52
14 Catholic                       2.96
15 Protestant                     3.15
```

```
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`.

```
tvdat$relig
```

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

```
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
```

```
# A tibble: 15 x 2
   relig                    tvhours_mean
   <fct>                           <dbl>
 1 No answer                        2.72
 2 Don't know                       4.62
 3 Inter-nondenominational          2.87
 4 Native american                  3.46
 5 Christian                        2.79
 6 Orthodox-christian               2.42
 7 Moslem/islam                     2.44
 8 Other eastern                    1.67
 9 Hinduism                         1.89
10 Buddhism                         2.38
11 Other                            2.73
12 None                             2.71
13 Jewish                           2.52
14 Catholic                         2.96
15 Protestant                       3.15
```

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

tvdat
```

```
# A tibble: 15 x 2
   relig                    tvhours_mean
   <fct>                           <dbl>
 1 No answer                        2.72
```

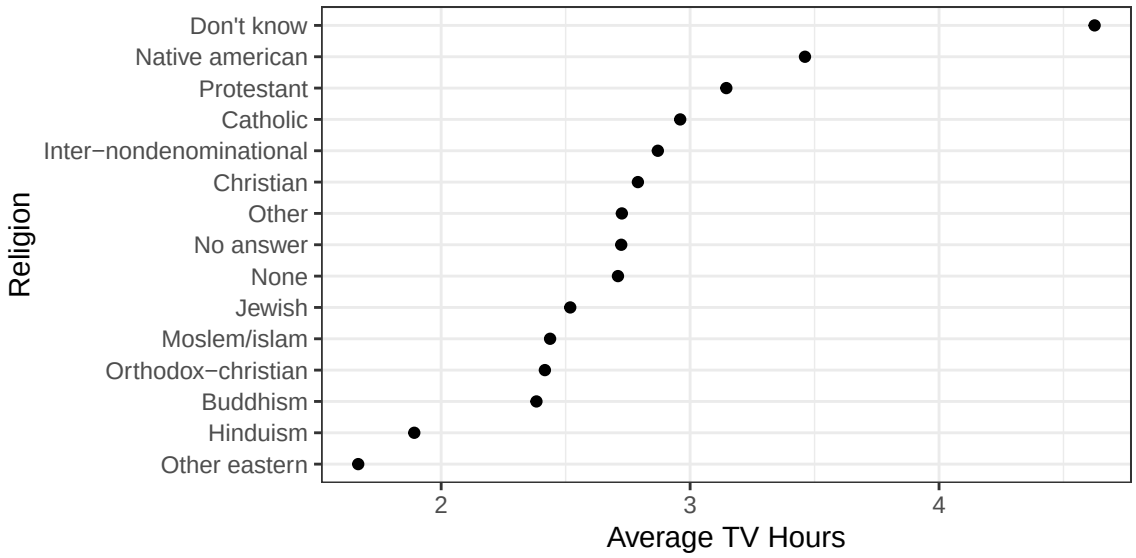
```
 2 Don't know                     4.62
 3 Inter-nondenominational        2.87
 4 Native american                3.46
 5 Christian                      2.79
 6 Orthodox-christian             2.42
 7 Moslem/islam                   2.44
 8 Other eastern                  1.67
 9 Hinduism                       1.89
10 Buddhism                       2.38
11 Other                          2.73
12 None                           2.71
13 Jewish                         2.52
14 Catholic                       2.96
15 Protestant                     3.15
```

```
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"
```

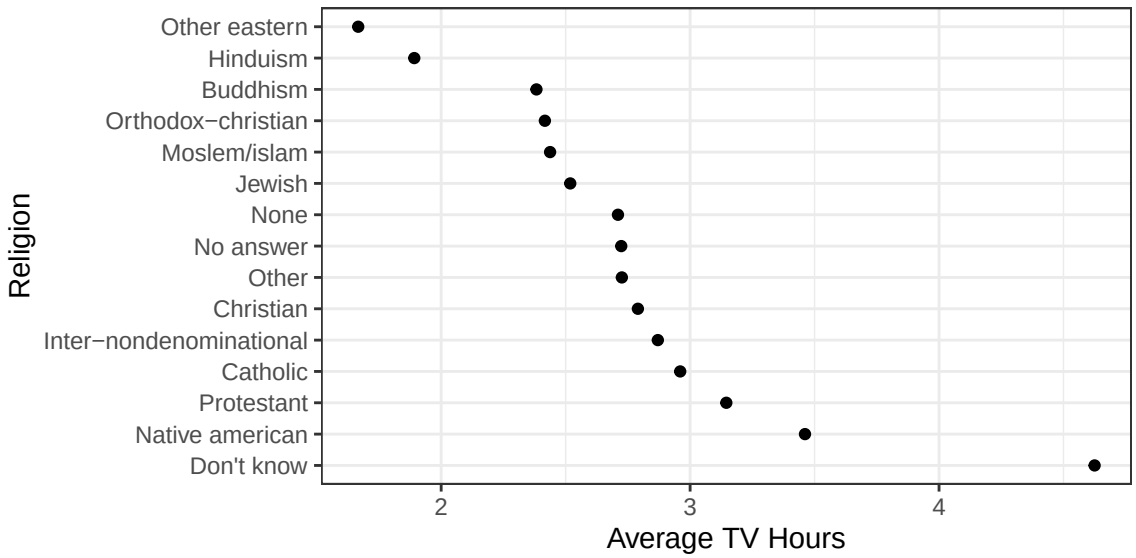
- The plot now reorders the y-axis according to the new level order.

```
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.

```
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"
```

```
    ## 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: 1 unknown level in `f`: Cthulhuism
```

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

- **Exercise**: Reorder the levels of the `partyid` variable so that the levels are in alphabetical order.

```
glimpse(gss_cat)
```

```
Rows: 21,483
Columns: 9
$ year    <int> 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 2000, 20~
$ marital <fct> Never married, Divorced, Widowed, Never married, Divorced, Mar~
$ age     <int> 26, 48, 67, 39, 25, 25, 36, 44, 44, 47, 53, 52, 52, 51, 52, 40~
$ race    <fct> White, White, White, White, White, White, White, White, White,~
$ rincome <fct> $8000 to 9999, $8000 to 9999, Not applicable, Not applicable, ~
$ partyid <fct> "Ind,near rep", "Not str republican", "Independent", "Ind,near~
$ relig   <fct> Protestant, Protestant, Protestant, Orthodox-christian, None, ~
$ denom   <fct> "Southern baptist", "Baptist-dk which", "No denomination", "No~
$ tvhours <int> 12, NA, 2, 4, 1, NA, 3, NA, 0, 3, 2, NA, 1, NA, 1, 7, NA, 3, 3~
```

```r
unique(gss_cat$partyid)
```

```
 [1] Ind,near rep       Not str republican Independent        Not str democrat
 [5] Strong democrat    Ind,near dem       Strong republican  Other party
 [9] No answer          Don't know
10 Levels: No answer Don't know Other party ... Strong democrat
```

```r
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"
```

```r
gss_cat %>%
      mutate(partyid = fct_relevel(partyid, sort(levels(partyid))))
```

```
# A tibble: 21,483 x 9
    year marital         age race  rincome        partyid    relig denom tvhours
   <int> <fct>         <int> <fct> <fct>          <fct>      <fct> <fct>   <int>
 1  2000 Never married    26 White $8000 to 9999  Ind,near ~ Prot~ Sout~      12
 2  2000 Divorced         48 White $8000 to 9999  Not str r~ Prot~ Bapt~      NA
 3  2000 Widowed          67 White Not applicable Independe~ Prot~ No d~       2
 4  2000 Never married    39 White Not applicable Ind,near ~ Orth~ Not ~       4
 5  2000 Divorced         25 White Not applicable Not str d~ None  Not ~       1
 6  2000 Married          25 White $20000 - 24999 Strong de~ Prot~ Sout~      NA
 7  2000 Never married    36 White $25000 or more Not str r~ Chri~ Not ~       3
 8  2000 Divorced         44 White $7000 to 7999  Ind,near ~ Prot~ Luth~      NA
 9  2000 Married          44 White $25000 or more Not str d~ Prot~ Other       0
10  2000 Married          47 White $25000 or more Strong re~ Prot~ Sout~       3
# i 21,473 more rows
```

- **Exercise**: Move the `"Not applicable"` level to the front in the `rincome` variable.

```r
levels(gss_cat$rincome)
```

```
 [1] "No answer"      "Don't know"     "Refused"        "$25000 or more"
 [5] "$20000 - 24999" "$15000 - 19999" "$10000 - 14999" "$8000 to 9999"
 [9] "$7000 to 7999"  "$6000 to 6999"  "$5000 to 5999"  "$4000 to 4999"
[13] "$3000 to 3999"  "$1000 to 2999"  "Lt $1000"       "Not applicable"
```

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"            "Other party"
 [4] "Republican, strong"    "Republican, weak"      "Independent, near rep"
 [7] "Independent"           "Independent, near dem" "Democrat, weak"
[10] "Democrat, strong"
```

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

```
      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"
```

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

```
levels(gss_cat$marital)
```

```
[1] "No answer"     "Never married" "Separated"     "Divorced"
[5] "Widowed"       "Married"
```

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"))
fc
```

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

```
    fct_count(fc)
```

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

```
fc
```

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

```
fct_infreq(fc)
```

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

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

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