Lecture 19: Factors

STAT 450, Fall 2021

Reading: Chapter 15 from *R for Data Science*

Useful reference: https://forcats.tidyverse.org/

In R, factors are used to work with categorical variables. Recall that a categorical variable takes on values that fall into distinct categories. Some examples are gender, education level, and political affiliation.

We will use the `forcats` package, which is a core tidyverse package that provides tools for working with factors.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Creating Factors

```
x <- c("low", "medium", "medium", "high", "low", "high" )
class(x)
```

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

By default, R sorts alphabetically, which is not that useful here.

```
sort(x)
```

```
## [1] "high"   "high"   "low"    "low"    "medium" "medium"
```

We can use the `factor()` function to convert the character vector `x` into a factor. The ordering of the categories is specified by the `levels` argument.

```
y <- factor(x, levels = c("low", "medium", "high"))
class(y)
```

```
## [1] "factor"
```

```
sort(y)
```

```
## [1] low    low    medium medium high   high
## Levels: low medium high
```

General Social Survey

The `gss_cat` data frame contains data from the General Social Survey, which is a long-running sociological survey conducted by the National Opinion Research Center at the University of Chicago.

```
gss_cat
```

```
## # 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,nea~ Prote~ South~      12
##  2  2000 Divorced         48 White $8000 to 9999  Not str~ Prote~ Bapti~      NA
##  3  2000 Widowed          67 White Not applicable Indepen~ Prote~ No de~       2
##  4  2000 Never married    39 White Not applicable Ind,nea~ Ortho~ Not a~       4
##  5  2000 Divorced         25 White Not applicable Not str~ None   Not a~       1
##  6  2000 Married          25 White $20000 - 24999 Strong ~ Prote~ South~      NA
##  7  2000 Never married    36 White $25000 or more Not str~ Chris~ Not a~       3
##  8  2000 Divorced         44 White $7000 to 7999  Ind,nea~ Prote~ Luthe~      NA
##  9  2000 Married          44 White $25000 or more Not str~ Prote~ Other        0
## 10  2000 Married          47 White $25000 or more Strong ~ Prote~ South~       3
## # ... with 21,473 more rows
```

Consider the categorical variable `race`, which is represented as a factor. The `levels()` function can be used to extract the different categories:

```
levels(gss_cat$race)
```

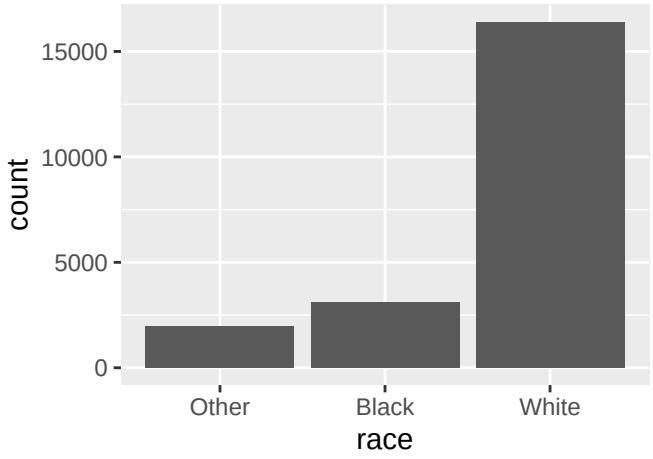
```
## [1] "Other"          "Black"          "White"          "Not applicable"
```

We can also make a table and bar plot displaying the number of respondents in each race category:

```
gss_cat %>%
  count(race)
```

```
## # A tibble: 3 x 2
##   race      n
##   <fct> <int>
## 1 Other  1959
## 2 Black  3129
## 3 White 16395
```

```
ggplot(gss_cat, aes(race)) + geom_bar()
```



By default `ggplot2` will drop any levels that don't have values. That's why `Not applicable` does not show up in the bar plot.

Exercise 1: Explore the distribution of `rincome` (reported income). What makes the default bar plot difficult to understand? How could you improve the plot?

Modifying factor order

It is often useful to change the order of the levels of a factor for data visualization.

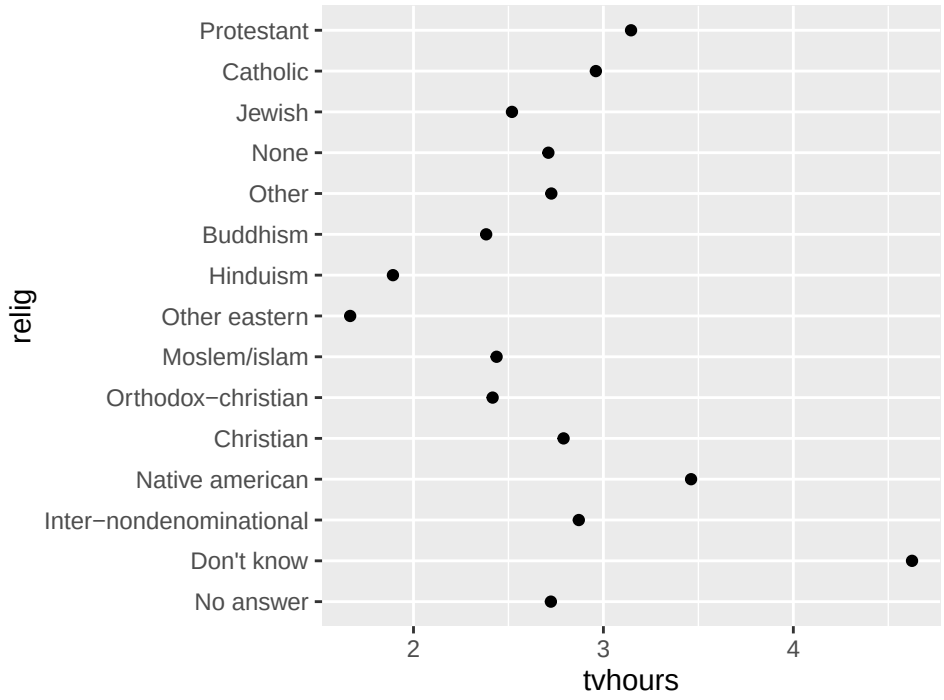
`fct_reorder()`

Suppose we want to explore the average number of hours per day spent watching TV across religions:

```
relig_summary <- gss_cat %>%
  group_by(relig) %>%
  summarise(
    n = n(),
    tvhours = mean(tvhours, na.rm = TRUE)
  )
relig_summary
```

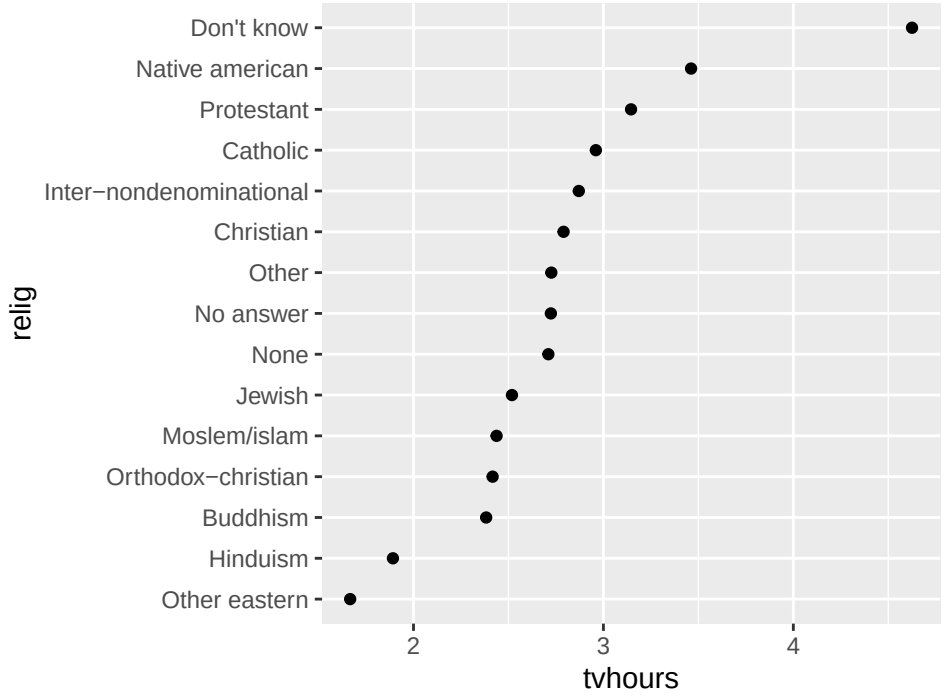
```
## # A tibble: 15 x 3
##    relig                      n tvhours
##    <fct>                  <int>   <dbl>
##  1 No answer                 93    2.72
##  2 Don't know                15    4.62
##  3 Inter-nondenominational  109    2.87
##  4 Native american           23    3.46
##  5 Christian                689    2.79
##  6 Orthodox-christian        95    2.42
##  7 Moslem/islam             104    2.44
##  8 Other eastern             32    1.67
##  9 Hinduism                  71    1.89
## 10 Buddhism                 147    2.38
## 11 Other                    224    2.73
## 12 None                    3523    2.71
## 13 Jewish                   388    2.52
## 14 Catholic                5124    2.96
## 15 Protestant             10846    3.15
```

```
ggplot(relig_summary, aes(tvhours, relig)) + geom_point()
```



We can improve this plot by using the function `fct_reorder()` to reorder the levels of `relig` according to the average number of hours per day spent watching TV:

```
ggplot(relig_summary, aes(tvhours, fct_reorder(relig, tvhours))) +
  geom_point() + ylab("relig")
```

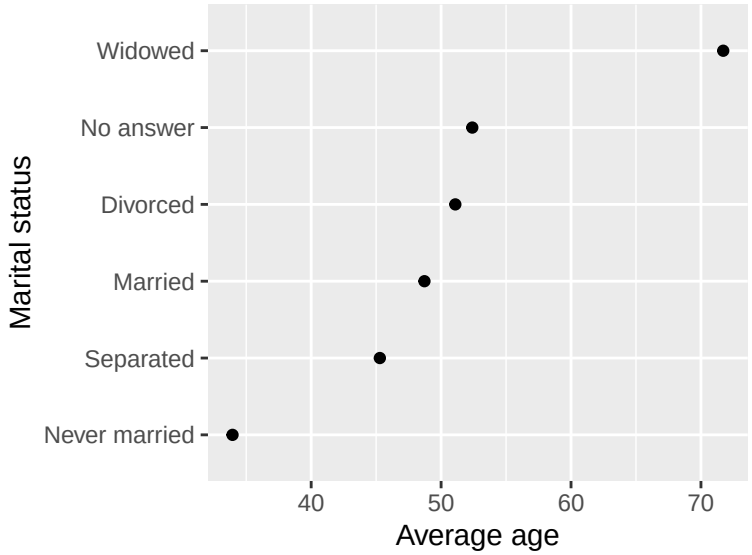


The following code which uses `mutate()` will create the same graph:

```
relig_summary %>%
  mutate(relig = fct_reorder(relig, tvhours)) %>%
ggplot(aes(tvhours, relig)) +
  geom_point()
```

This code is somewhat easier to read in my view, since it clearly shows the two steps here: (1) reorder the levels of `relig` according to `tvhours`, the average number of hours spent watching tv; and (2) make a plot with `tvhours` on the x-axis and `relig` on the y-axis.

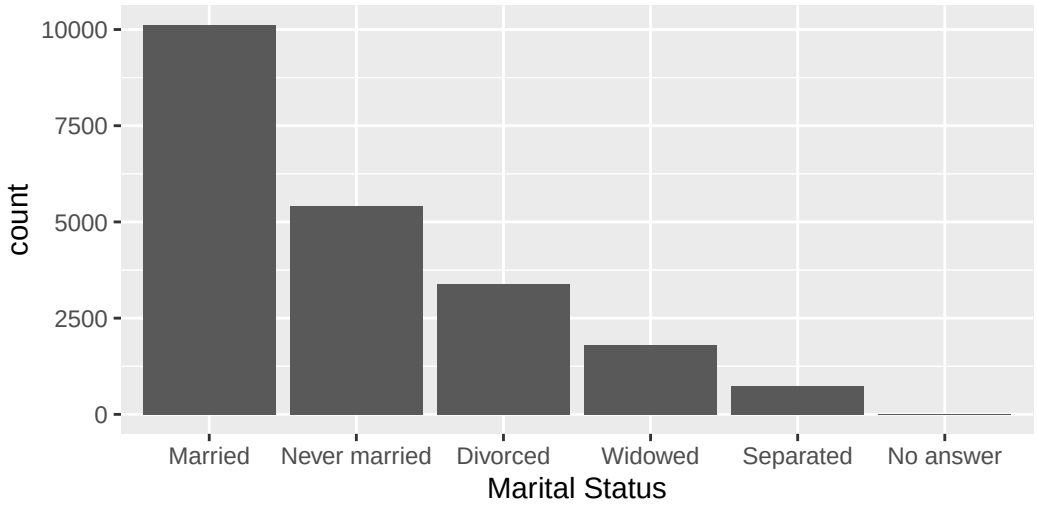
Exercise 2: Use `group_by()` and `summarise()` to compute the average age for each category of `marital`. Call the data frame with the grouped summaries `marital_summary`. Then recreate the R code that makes the graph below.



`fct_infreq()`

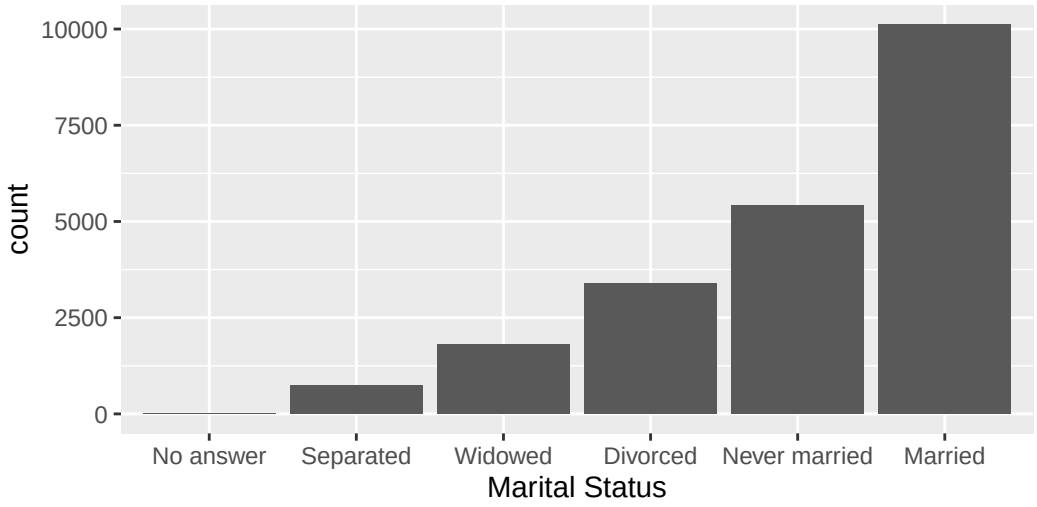
`fct_infreq()` orders the levels of a factor by frequency (number of observations in each level).

```
ggplot(gss_cat, aes(fct_infreq(marital))) +
  geom_bar() + xlab("Marital Status")
```



You can combine with `fct_rev()` to reverse the order.

```
ggplot(gss_cat, aes(fct_rev(fct_infreq(marital)))) +
  geom_bar() + xlab("Marital Status")
```



The following code which uses `mutate()` and the `%>%` pipe operator will create the same graph:

```
gss_cat %>%
  mutate(marital = marital %>% fct_infreq() %>% fct_rev()) %>%
  ggplot(aes(marital)) +
    geom_bar()
```

Modifying factor levels

`fct_recode()`

Use `fct_recode()` to change the names of the levels. For example:

```
gss_cat %>% count(partyid)
```

```
## # A tibble: 10 x 2
##    partyid               n
##    <fct>             <int>
##  1 No answer           154
##  2 Don't know            1
##  3 Other party         393
##  4 Strong republican  2314
##  5 Not str republican 3032
##  6 Ind,near rep       1791
##  7 Independent        4119
##  8 Ind,near dem       2499
##  9 Not str democrat   3690
## 10 Strong democrat    3490
```

The levels for `partyid` are inconsistent, and can be improved.

```
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"
  )) %>%
  count(partyid)
```

```
## # A tibble: 10 x 2
##    partyid                  n
##    <fct>                <int>
##  1 No answer              154
##  2 Don't know               1
##  3 Other party            393
##  4 Republican, strong    2314
##  5 Republican, weak      3032
##  6 Independent, near rep 1791
##  7 Independent           4119
##  8 Independent, near dem 2499
##  9 Democrat, weak        3690
## 10 Democrat, strong      3490
```

`fct_recode()` will not alter any levels that aren't explicitly mentioned.

`fct_collapse()`

Use `fct_collapse()` to combine levels. For example:

```
gss_cat %>%
  mutate(partyid = fct_collapse(partyid,
    other = c("No answer", "Don't know", "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")
  )) %>%
  count(partyid)
```

```
## # A tibble: 4 x 2
##   partyid     n
##   <fct>   <int>
## 1 other     548
## 2 rep      5346
## 3 ind      8409
## 4 dem      7180
```