QTM 151

Lab 04 – forcats

Umberto Mignozzetti
Summer

Recap

We learned:

- `qplot`: quick way to make ggplot graphs.
- `ggplotly`: transform ggplot objects into nice plotly viz.
- `plot_ly`: create nice plotly graphs.
- `dplyr *_join` methods: joining data

Great job!!

Do you have any questions about any of these contents?

Today we are going to talk about **forcats** (package for categorical variables)

Today's Class

We have a **quiz** for today's class, due by the next class.

Are you done with DataCamp? Is it being helpful?

What about the final project? Are you in contact with your group?

Our GitHub page is: https://github.com/umbertomig/qtm151

Getting Started

Getting Started: loading packages

```
# Loading tidyverse
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyv

## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.2     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse_c
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Loading data - GSS

```
# Loading the GSS Cat and GSS
gss_cat ← read_csv('https://raw.githubusercontent.com/umbertomig/
```

```
##
## ── Column specification ────────────────────────────────────────
## cols(
##   year = col_double(),
##   marital = col_character(),
##   age = col_double(),
##   race = col_character(),
##   rincome = col_character(),
##   partyid = col_character(),
##   relig = col_character(),
##   denom = col_character(),
##   tvhours = col_double()
## )
```

forcats

forcats

Provide tools to work with categorical data. The methods we will use in here are:

- `fct_reorder()`: reorder levels in a categorical variable
- `fct_relevel()`: move the position of a particular level
- `fct_infreq()`: reorder levels in increasing frequency
- `fct_recode()`: recode factor variable
- `fct_collapse()`: collapse levels in categorical variable
- `fct_lump()`: lump together small groups
- `case_when()`: useful to recode data

fct_reorder

fct_reorder

Suppose we want to study the relationship between religion and time watching tv.

This graph here doesn't help much:

```
relig_summary ← gss_cat %>%
  group_by(relig) %>%
  summarise(tvhours=mean(tvhours, na.rm = T))

ggplot(relig_summary, aes(x=tvhours, y=relig)) +
  geom_point()
```

fct_reorder

Reordering the levels of a factor using *fct_reorder()* helps with the interpretation.

fct_reorder() takes three arguments:

- f, the factor whose levels you want to modify.
- x, a numeric vector that you want to use to reorder the levels.
- Optionally, fun, a function used if there are multiple values of x for each value of f. The default value is median.

```
ggplot(relig_summary, aes(x=tvhours, y=fct_reorder(relig, tvhours))) +
  geom_point()
```

Your turn: plot the marital status by frequency (use the function counts). Then reorder it.

fct_relevel

fct_relevel

In the religions x tv hours plot, note that the "Don't know" is the largest category.

It is also not very informative. We can easily send it to the bottom:

```
ggplot(relig_summary,
       aes(x = tvhours,
           y = fct_relevel(relig, "Don't know", after=0))) +
    geom_point()
```

fct_relevel

Can we do ascending, and place the "Don't know" in the bottom?

Yes!

```
ggplot(relig_summary,
       aes(x=tvhours,
           y = fct_reorder(relig, tvhours) %>%
             fct_relevel("Don't know", after=0))) +
  geom_point()
```

Your turn: plot the party id by frequency (use the function counts). Then reorder it. Then place "No answer" and "Independent" at the bottom.

fct_infreq

fct_infreq

You can use `fct_infreq()` to order levels in increasing frequency:.

- This is the simplest type of reordering because it does not need any extra variables.

You may want to combine with `fct_rev()`.

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

fct_infreq + fct_rev (elegant)

And that's what happens when we combine both:

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

Your turn: plot the race by frequency.

fct_recode

fct_recode

Recode a categorical variable is always painful, regardless of the statistical software.

Luckly, the people that wrote forcats made it as easy as it can be by creating the `fct_recode` function.

Check this plot. TV hours by party affiliation:

```
gss_cat %>%
  drop_na(tvhours) %>%
  group_by(partyid) %>%
  summarise(meantv=mean(tvhours)) %>%
  ggplot(aes(x=meantv, y=fct_reorder(partyid, meantv)))+
  geom_point()
```

fct_recode

Note that we can combine some low-information categories together as `others`:

```
gss_cat %>%
  drop_na(tvhours) %>%
  mutate(partyidnew = 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",
    "Other"                 = "No answer",
    "Other"                 = "Don't know",
    "Other"                 = "Other party")) %>%
  group_by(partyidnew) %>%
  summarise(meantvhours = mean(tvhours)) %>%
  ggplot(aes(x=meantvhours,
```

fct_collapse

fct_collapse

`fct_collapse` is good to put several factor levels together.

Look at this code for party id frequency:

```
gss_cat %>%
  mutate(partyidnew = 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(partyidnew)
```

Your turn: collapse the gss `wrkstat` variable to simplify it.

fct_lump

fct_lump

`fct_lump` aggregates the small-frequency levels together.

The default is the most frequent + others:

```
gss_cat %>%
  mutate(relignew = fct_lump(relig)) %>%
  count(relignew)
```

fct_lump

If we want the five most frequent values + others:

```
gss_cat %>%
  mutate(relig = fct_lump(relig, n=5)) %>%
  count(relig)
```

Your turn: count the levels in the `gss_cat` `denom` variable. Find a nice way to display it.

case_when

case_when

`case_when` is useful for recoding variables. Look at the example below:

```
mtcars %>%
  mutate (
    gear_char =
      case_when(
        gear==3 ~ "three",
        gear==4 ~ "four",
        gear==5 ~ "five"
      )
  )
```

Your turn: for `gss_cat`, create a plot to look at how average age varies across income level (rincome)

Questions?

See you next class!