Refresher on R essentials

PSCI 2301: Quantitative Political Science II

Prof. Brenton Kenkel

brenton.kenkel@gmail.com

Vanderbilt University

January 8, 2025

Today's agenda

1. Getting data into R
2. Manipulating and cleaning data with tidyverse
3. Data visualization with ggplot

Before we go further: Packages

All R code I write in this course will assume you've got the tidyverse R package installed and loaded.

```
library("tidyverse")
```

If that gives you an error message like...

```
Error in library("tidyverse") : there is no package called ‘tidyverse’
```

...then you need to install the package by running

```
install.packages("tidyverse")
```

Getting data into R

Data from the public Internet

If you have a direct link to a CSV file, you can plug it into `read_csv()`.

Remember to put the URL inside quote marks.

```
df_lottery <- read_csv("https://data.ny.gov/api/views/5xaw-6ayf/rows.csv")
print(df_lottery)
```

```
# A tibble: 2,360 × 4
  `Draw Date` `Winning Numbers` `Mega Ball` Multiplier
  <chr>       <chr>             <chr>       <chr>
1 09/25/2020  20 36 37 48 67    16          02
2 09/29/2020  14 39 43 44 67    19          03
3 10/02/2020  09 38 47 49 68    25          02
4 10/06/2020  15 16 18 39 59    17          03
5 10/09/2020  05 11 25 27 64    13          02
# ℹ 2,355 more rows
```

Data from a downloaded file

Often you can't `read_csv()` directly from a URL:

- Files behind logins, e.g. on the course Brightspace
- Files within a zipped directory
- Files that somebody emailed you

In this situation, you need to be mindful of your **working directory**

Use `getwd()` to find out where R is looking for files, `setwd()` to change it

```
getwd()
```

```
[1] "/home/brenton/Dropbox/courses/qps2/slides/01_02_r_refresher"
```

Data from a downloaded file: Practice

1. Create a directory somewhere you can find it
2. Download the `anes2020.csv` file from Brightspace and put it there
3. Set as R's working directory
   - Windows: `setwd('C:/path/to/directory')`
   - Mac: `setwd('~/path/to/directory')`
   - … or just navigate in RStudio and set it that way
4. Check that `df_anes <- read_csv("anes2020.csv")` works

```r
df_anes <- read_csv("anes2020.csv")

print(df_anes)
```

```
# A tibble: 8,280 × 32
     id state       female  lgbt race    age
  <dbl> <chr>        <dbl> <dbl> <chr> <dbl>
1     1 Oklahoma         0     0 Hisp…    46
2     2 Idaho            1     0 Asian    37
3     3 Virginia         1     0 White    40
4     4 Californ…        0     0 Asian    41
5     5 Colorado         0     0 Nati…    72
# ℹ 8,275 more rows
# ℹ 26 more variables: education <chr>,
#   employed <dbl>, hours_worked <dbl>, …
```

`read_csv` versus `read.csv`

I always use tidyverse's `read_csv` (w/ underscore) instead of R's built-in `read.csv` (w/ period)

- Automatically stores data frame as "tibble" ⇝ better output display
- Does *not* automatically encode text as "factor"
- Works faster + shows progress bar for large datasets

Manipulating data

Basics of data manipulation

Use $ to extract a single column

```
df_anes$age
```

```
 [1] 46 37 40 41 72 71 37 45 70 43 37 55 30 38 41 66 54 55 62 80 31 80 24 55 59
 [ reached getOption("max.print") -- omitted 8255 entries ]
```

Use square brackets [ ] to extract individual value(s)

```
df_anes$age[5]     # age of the 5'th row of the data
```

```
[1] 72
```

```
df_anes$age[1:10]  # first 10 ages in the data
```

```
 [1] 46 37 40 41 72 71 37 45 70 43
```

Useful data summaries

```
mean(df_anes$age, na.rm = TRUE)    # average/mean
```

```
[1] 51.58522
```

```
median(df_anes$age, na.rm = TRUE)  # median
```

```
[1] 52
```

```
sd(df_anes$age, na.rm = TRUE)      # standard deviation
```

```
[1] 17.20718
```

```
table(df_anes$race)                # counts of values -> can also do w/ summarize() or count()
```

```

          Asian           Black        Hispanic     Multiracial Native American 
            284             726             762             271             172 
          White 
           5963 
```

Reducing data by row or column

```r
filter(df_anes, age >= 75)              # by row
```

```
# A tibble: 793 × 32
     id state        female  lgbt race       age education employed hours_worked watch_tucker
  <dbl> <chr>         <dbl> <dbl> <chr>    <dbl> <chr>        <dbl>        <dbl>        <dbl>
1    20 Wisconsin         1     0 White       80 High sch…        0            0            0
2    22 California        1     0 Hispanic    80 Less tha…        0            0            0
3    47 Pennsylvania      0     0 White       79 Some col…        0            0            0
4    63 Tennessee         0     0 White       80 Graduate…        0            0            0
5    68 California        1     0 White       78 Graduate…        0            0            0
# ℹ 788 more rows
# ℹ 22 more variables: watch_maddow <dbl>, therm_biden <dbl>, therm_trump <dbl>, …
```

```r
select(df_anes, female, age, education)   # by column
```

```
# A tibble: 8,280 × 3
  female   age education
   <dbl> <dbl> <chr>
1      0    46 Bachelor's degree
2      1    37 Some college
3      1    40 High school
4      0    41 Some college
5      0    72 Graduate degree
# ℹ 8,275 more rows
```

Chaining commands with the pipe

```r
df_anes |>
  filter(age >= 75) |>
  select(female, age, education)
```

```
# A tibble: 793 × 3
  female   age education
   <dbl> <dbl> <chr>
1      1    80 High school
2      1    80 Less than high school
3      0    79 Some college
4      0    80 Graduate degree
5      1    78 Graduate degree
# ℹ 788 more rows
```

Different pipes

I use R's built-in pipe `|>`. Online you'll find a lot of code using the tidyverse pipe `%>%`. Both are fine and do essentially the same thing.

Changing and adding columns

```r
df_anes |>
  mutate(female = if_else(female == 1, "yes", "no"),
         age_in_days = age * 365,
         employment_type = case_when(
           hours_worked == 0 ~ "unemployed",
           hours_worked < 32 ~ "part-time",
           hours_worked >= 32 ~ "full-time"
         )) |>
  relocate(female, age_in_days, employment_type)  # put these cols first
```

```
# A tibble: 8,280 × 34
  female age_in_days employment_type    id state       lgbt race        age education employed
  <chr>        <dbl> <chr>           <dbl> <chr>      <dbl> <chr>     <dbl> <chr>        <dbl>
1 no           16790 full-time           1 Oklahoma       0 Hispanic     46 Bachelor…        1
2 yes          13505 full-time           2 Idaho          0 Asian        37 Some col…        1
3 yes          14600 unemployed          3 Virginia       0 White        40 High sch…        0
4 no           14965 full-time           4 California     0 Asian        41 Some col…        1
5 no           26280 unemployed          5 Colorado       0 Native …     72 Graduate…        0
# ℹ 8,275 more rows
# ℹ 24 more variables: hours_worked <dbl>, watch_tucker <dbl>, watch_maddow <dbl>, …
```

Summaries by group

```r
df_anes |>
  group_by(race) |>
  summarize(n_respondents = n(),
            avg_trump_feeling = mean(therm_trump, na.rm = TRUE),
            sd_trump_feeling = sd(therm_trump, na.rm = TRUE))
```

```
# A tibble: 7 × 4
  race            n_respondents avg_trump_feeling sd_trump_feeling
  <chr>                   <int>             <dbl>            <dbl>
1 Asian                     284              34.0             36.1
2 Black                     726              15.0             25.2
3 Hispanic                  762              32.2             36.7
4 Multiracial               271              34.8             38.8
5 Native American           172              42.2             38.1
6 White                    5963              45.0             41.2
7 <NA>                      102              41.3             38.3
```

Making changes stick

R commands almost never change a data frame in memory. The results of `filter()`, `select()`, `mutate()`, etc., will disappear unless you use `<-` to overwrite the original data frame or create a new one.

```
df_anes_women <- df_anes |>
  filter(female == 1)
```

```
df_anes
```

```
# A tibble: 8,280 × 32
     id state      female  lgbt race        age
  <dbl> <chr>       <dbl> <dbl> <chr>     <dbl>
1     1 Oklahoma        0     0 Hispanic     46
2     2 Idaho           1     0 Asian        37
3     3 Virginia        1     0 White        40
4     4 California      0     0 Asian        41
5     5 Colorado        0     0 Native …     72
# ℹ 8,275 more rows
# ℹ 26 more variables: education <chr>,
#   employed <dbl>, hours_worked <dbl>, …
```

```
df_anes_women
```

```
# A tibble: 4,450 × 32
     id state     female  lgbt race    age
  <dbl> <chr>      <dbl> <dbl> <chr> <dbl>
1     2 Idaho          1     0 Asian    37
2     3 Virginia       1     0 White    40
3     6 Texas          1     0 White    71
4     7 Wisconsin      1     0 White    37
5     8 <NA>           1     0 White    45
# ℹ 4,445 more rows
# ℹ 26 more variables: education <chr>,
#   employed <dbl>, hours_worked <dbl>, …
```

Other helpful data manipulation commands

- `case_match()` to code one column based on values of another
- `arrange()` for reordering rows
- `pivot_wider()` and `pivot_longer()` for reshaping data frames
- `left_join()` for merging data frames
- `group_by() |> mutate()` to add columns based on group-level calculations

Additional info on these commands

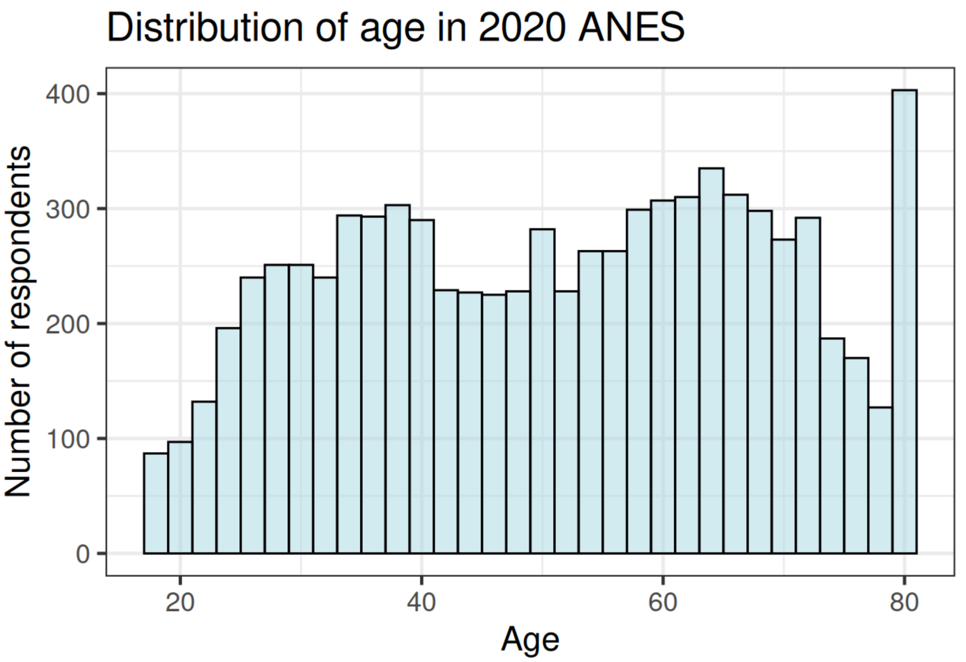
- The “Data Wrangling” notes from PSCI 2300 posted to Brightspace
- Lecture notes from my graduate stats class
- “Data Transformation” and “Data Tidying” chapters of *R for Data Science*

Data visualization

Visualizing a single variable

Continuous variables: Histogram

```r
ggplot(df_anes, aes(x = age)) +
  geom_histogram(color = "black",
                 fill = "lightblue",
                 alpha = 0.5,
                 binwidth = 2) +
  labs(x = "Age",
       y = "Number of respondents",
       title = "Distribution of age in 2020 ANES")
```



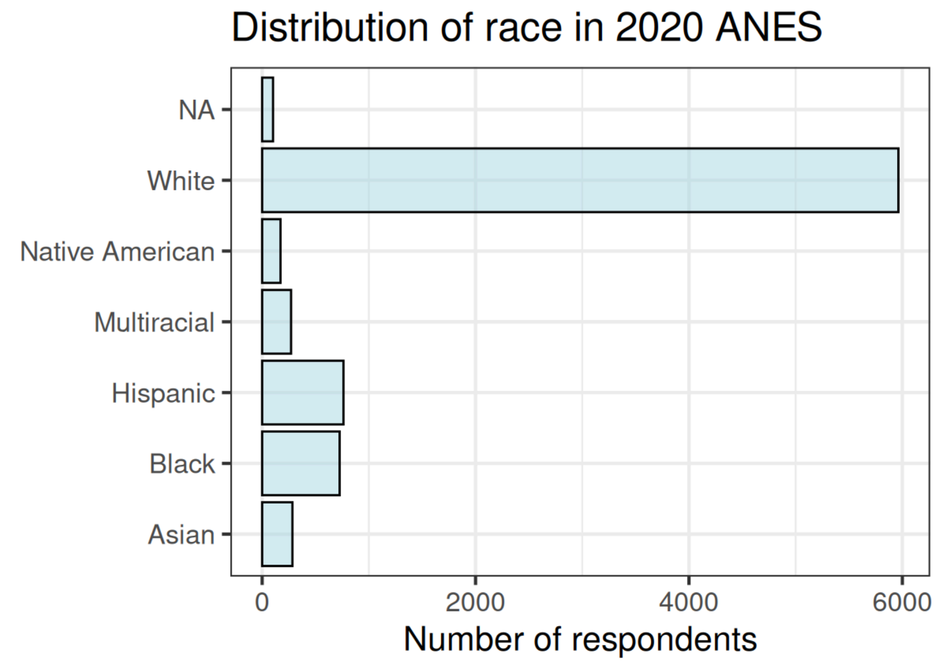
⚠ ggplot syntax

ggplot commands are separated by addition `+`, not the pipe `|>`.

Visualizing a single variable

Categorical variables: Bar chart

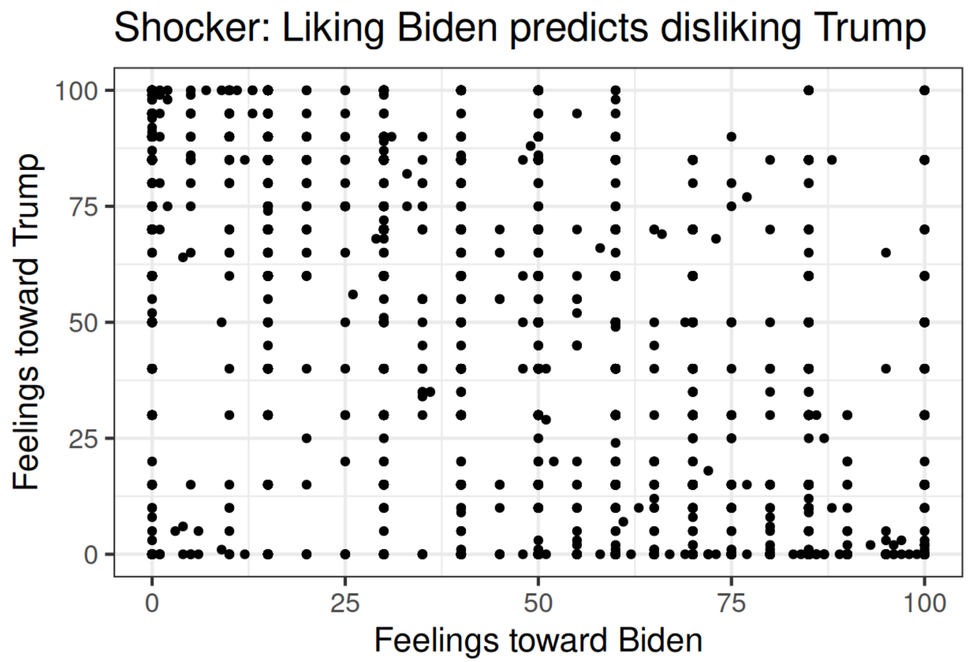
```
ggplot(df_anes, aes(y = race)) +
  geom_bar(color = "black",
           fill = "lightblue",
           alpha = 0.5) +
  labs(x = "Number of respondents",
       y = "",
       title = "Distribution of race in 2020 ANES")
```



Visualizing relationships

Two continuous variables: Scatterplot

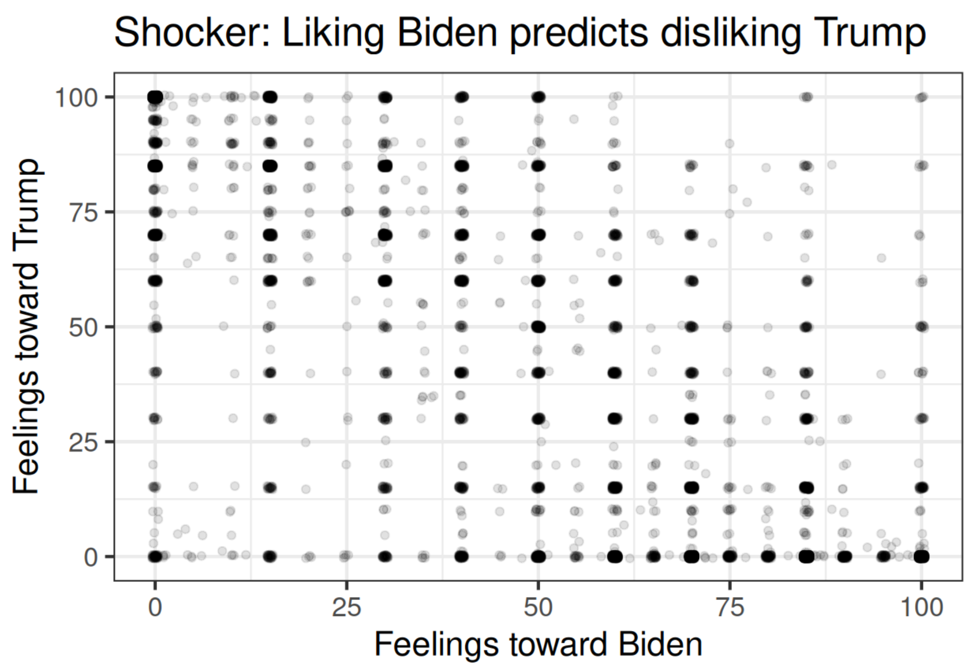
```r
ggplot(df_anes,
        aes(x = therm_biden, y = therm_trump))
  geom_point() +
  labs(x = "Feelings toward Biden",
        y = "Feelings toward Trump",
        title = "Shocker: Liking Biden predicts
```



Visualizing relationships

Two continuous variables: Scatterplot

```
ggplot(df_anes,
       aes(x = therm_biden, y = therm_trump)) +
  geom_point(position = "jitter",
             alpha = 0.1) +
  labs(x = "Feelings toward Biden",
       y = "Feelings toward Trump",
       title = "Shocker: Liking Biden predicts
```



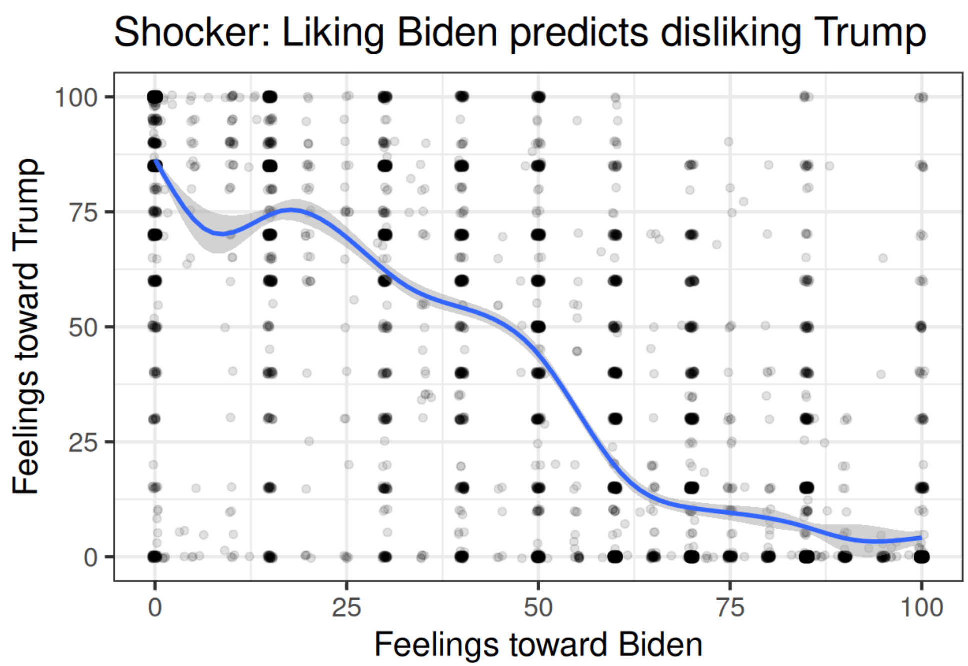
Jitter and transparency

When data is clumpy with lots of overlapping values, jitter the point locations and/or make points semi-transparent to see relationships better.

Visualizing relationships

Two continuous variables: Scatterplot

```r
ggplot(df_anes,
       aes(x = therm_biden, y = therm_trump))
  geom_point(position = "jitter",
             alpha = 0.1) +
  geom_smooth() +
  labs(x = "Feelings toward Biden",
       y = "Feelings toward Trump",
       title = "Shocker: Liking Biden predicts
```



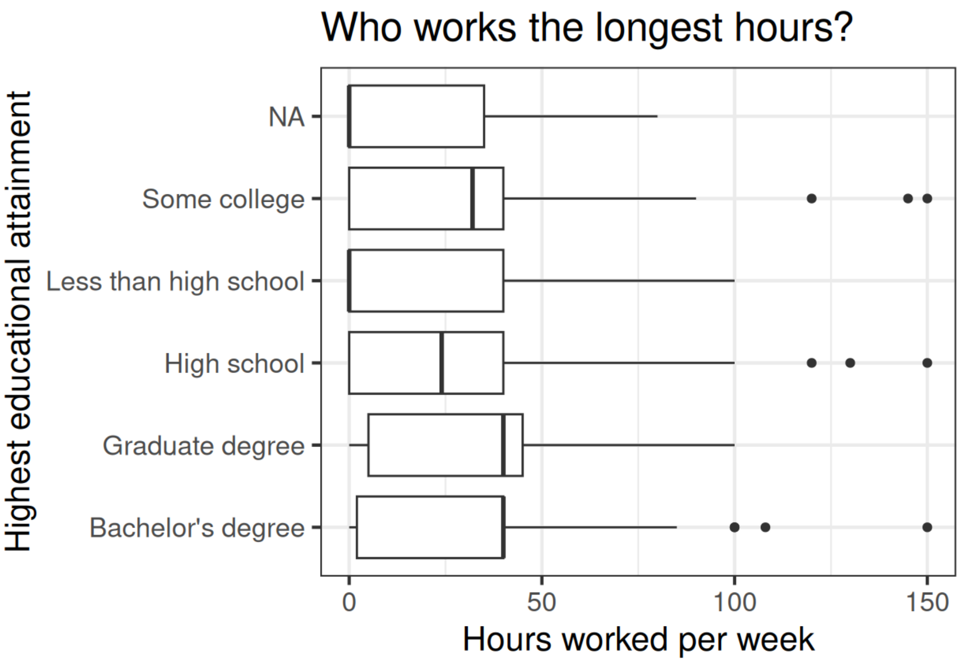
ⓘ Smoothing lines

Use `geom_smooth()` for a flexible trend line, or `geom_smooth(method = "lm")` for the linear regression line.

Visualizing relationships

Continuous and categorical variable: Box plot

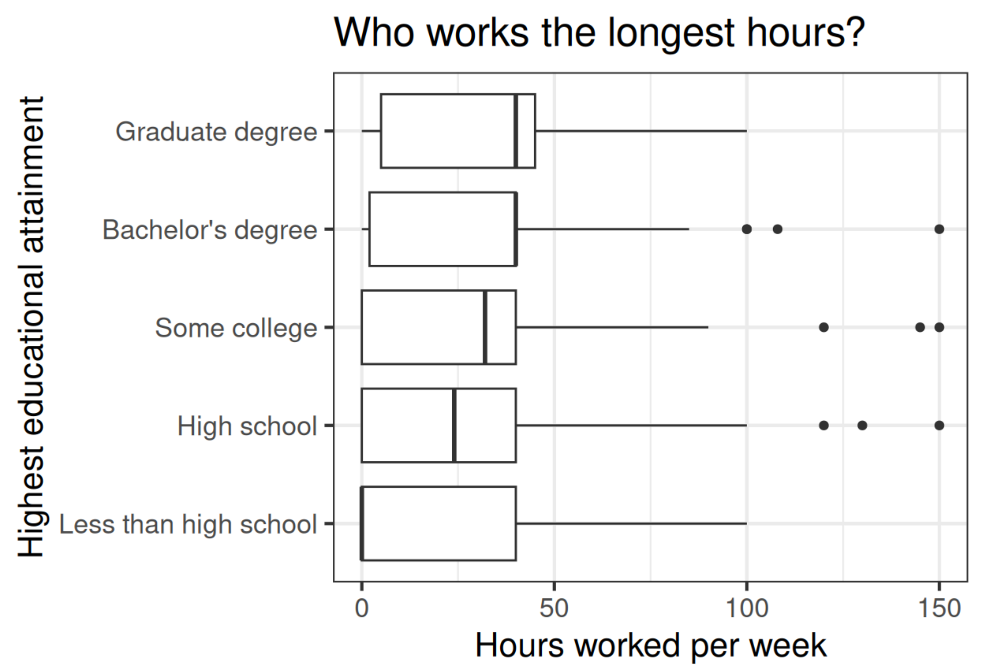
```
ggplot(df_anes, aes(x = hours_worked, y = educ
  geom_boxplot() +
  labs(x = "Hours worked per week",
       y = "Highest educational attainment",
       title = "Who works the longest hours?")
```



Visualizing relationships

Continuous and categorical variable: Box plot

```
ggplot(df_anes, aes(x = hours_worked, y = educ
  geom_boxplot() +
  scale_y_discrete(limits = c("Less than high
                              "High school",
                              "Some college",
                              "Bachelor's degr
                              "Graduate degree
  labs(x = "Hours worked per week",
       y = "Highest educational attainment",
       title = "Who works the longest hours?")
```



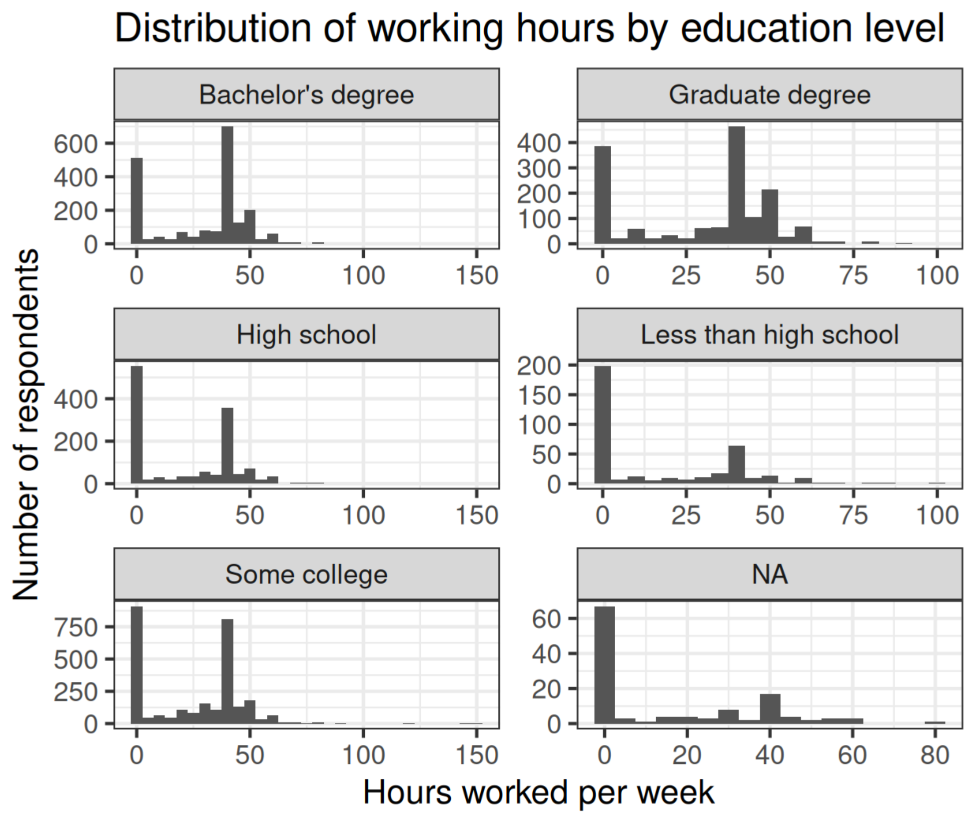
ⓘ Reordering a categorical variable

Use `scale_x_discrete(limits = c(...))` or `scale_y_discrete(limits = c(...))` to change the order the categories appear in.

Visualizing relationships

Continuous and categorical variable: Faceted histogram

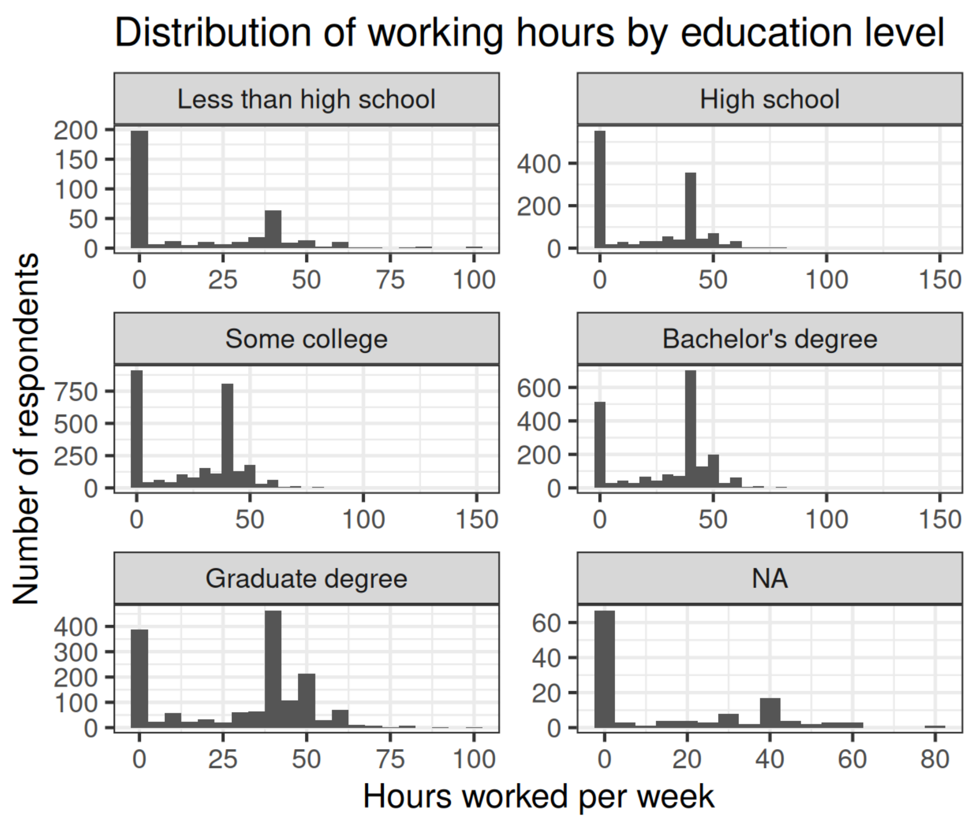
```r
ggplot(df_anes, aes(x = hours_worked)) +
  geom_histogram(binwidth = 5) +
  facet_wrap(~ education,
             scales = "free",
             ncol = 2) +
  labs(x = "Hours worked per week",
       y = "Number of respondents",
       title = "Distribution of working hours
```



Visualizing relationships

Continuous and categorical variable: Faceted histogram

```
ggplot(df_anes, aes(x = hours_worked)) +
  geom_histogram(binwidth = 5) +
  facet_wrap(~ fct_relevel(education,
                           "Less than high sch
                           "High school",
                           "Some college",
                           "Bachelor's degree"
             scales = "free",
             ncol = 2) +
  labs(x = "Hours worked per week",
       y = "Number of respondents",
       title = "Distribution of working hours
```



ⓘ Reordering facets

A bit fussier than reordering categories on an axis—use `fct_relevel()` within the call to `facet_wrap()`.

Visualizing relationships

Continuous and categorical variable: Bar chart summary

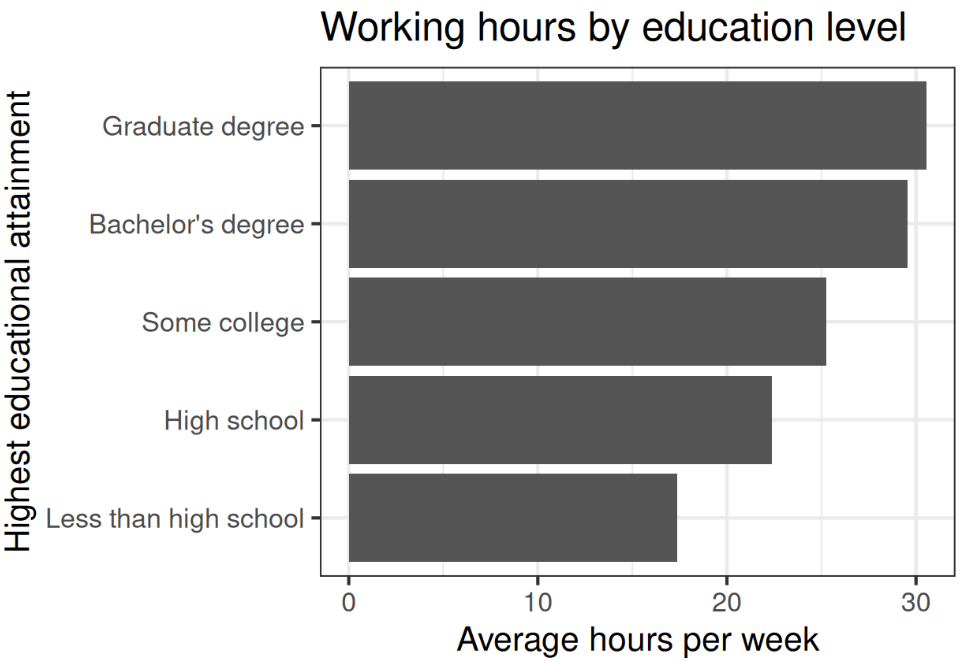
```r
df_anes |>
  group_by(education) |>
  summarize(avg_hours = mean(hours_worked,
                             na.rm = TRUE))
```

```
# A tibble: 6 × 2
  education              avg_hours
  <chr>                      <dbl>
1 Bachelor's degree           29.6
2 Graduate degree             30.6
3 High school                 22.4
4 Less than high school       17.4
5 Some college                25.2
6 <NA>                        15.9
```

Visualizing relationships

Continuous and categorical variable: Bar chart summary

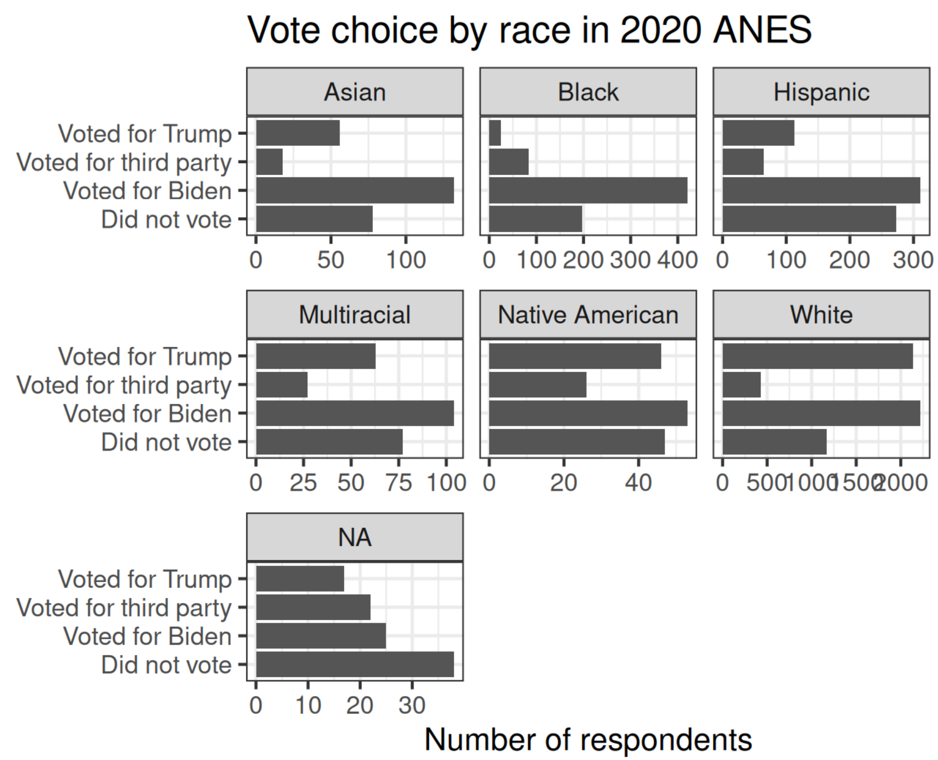
```
df_anes |>
  group_by(education) |>
  summarize(avg_hours = mean(hours_worked,
                             na.rm = TRUE)) |>
  ggplot(aes(x = avg_hours, y = education)) +
  geom_bar(stat = "identity") +
  scale_y_discrete(limits = c("Less than high
                              "High school",
                              "Some college",
                              "Bachelor's degr
                              "Graduate degree
  labs(title = "Working hours by education lev
       x = "Average hours per week",
       y = "Highest educational attainment")
```



Visualizing relationships

Two categorical variables: Faceted bar chart

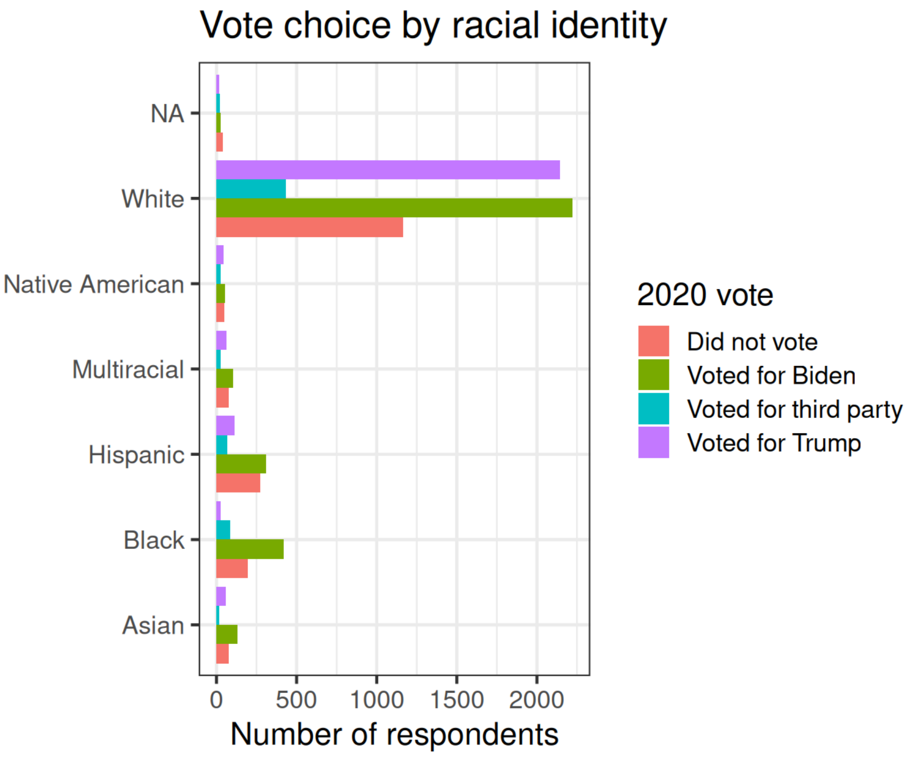
```r
ggplot(df_anes, aes(y = vote_type)) +
  geom_bar() +
  facet_wrap(~ race, scales = "free_x") +
  labs(x = "Number of respondents",
       y = "",
       title = "Vote choice by race in 2020 ANES")
```



Visualizing relationships

Two categorical variables: Dodged bar chart

```
ggplot(df_anes, aes(y = race)) +
  geom_bar(aes(fill = vote_type),
           position = "dodge") +
  labs(x = "Number of respondents",
       y = "",
       fill = "2020 vote",
       title = "Vote choice by racial identity")
```



Wrapping up

What we did today

1. Got data into R
    - File directly on web ⇝ `read_csv("https://url.com/file.csv")`
    - Otherwise ⇝ set working directory, save file there, `read_csv("file.csv")`
2. Manipulated data
    - Subset by row with `filter()`, by column with `select()`
    - Add or change columns with `mutate()`
    - Calculate summaries with `group_by()` and `summarize()`
    - Chain commands with the pipe `|>`
3. Visualized data with ggplot
    - Histograms and bar charts for one-variable summaries
    - Scatterplots, box plots, faceting for relationships

To do for next time

Next week's topic: Causal questions and research design

1. If anything from today was unfamiliar, practice with it
2. Read "Correlation, Causation, and Confusion" article
3. Read "Introduction to Causality" ebook chapter
4. Start thinking about topics you want to study in final project