Analysis of bootcamp survey

Rick Gilmore

2018-07-25 08:38:00

Contents

Goals

- Download and clean data from 2018 R Bootcamp Survey
- Visualize data
- Prepare reports in `ioslides_presentation`, `pdf_document`, and `word_document` formats

Preliminaries

Load required packages.

```
library(tidyverse)
library(googlesheets)
```

Load data and examine

The survey data are stored in a Google Sheet. We'll use the `googlesheets` package to open it and create a data frame. Documentation about the package can be found here.

There are some idiosyncrasies in using the `googlesheets` package in an R Markdown document because it requires interaction with the console, so I created a separate R script, `Get_bootcamp_googlesheet.R` to extract the survey data. If you try to execute the next chunk, it may give you an error, or it may ask you to allow `googlesheets` to access information in your Google profile.

```
survey_url <- "https://docs.google.com/spreadsheets/d/1-YBOiWUNN_9oxBhz221NFiyBOcwMfHziFeUiUvQwn7k/edit

bootcamp_by_url <- survey_url %>%
  googlesheets::extract_key_from_url() %>%
  googlesheets::gs_key()

bootcamp_sheets <- gs_ws_ls(bootcamp_by_url)

boot_data <- bootcamp_by_url %>%
  gs_read(bootcamp_sheets[1])

write_csv(boot_data, path=params$data_file_out)
```

This script downloads the data file saves it to a CSV under ../data/survey_2018.csv. We can then load this file.

I also created a test data file, `data/survey-test.csv` so I could see how everything worked before y'all filled out your responses. The `R/Make_test_survey.R` file shows how I did this. It's a great, reproducible practice to **simulate the data you expect**, then run it through your pipeline.

```
# Created test data set for testing.
survey <- read_csv(params$data_file_in)
```

```
## Parsed with column specification:
## cols(
##   Timestamp = col_datetime(format = ""),
##   R_exp = col_character(),
##   Banjo = col_integer(),
##   Psych_age_yrs = col_integer(),
##   Sleep_hrs = col_double(),
##   Fav_date = col_date(format = ""),
##   Crisis = col_character()
## )
```

```
# Or choose data from respondents
#survey <- read_csv(data_file_in)
```

```
survey
```

```
## # A tibble: 50 x 7
##    Timestamp           R_exp   Banjo Psych_age_yrs Sleep_hrs Fav_date
##    <dttm>              <chr>   <int>         <int>     <dbl> <date>
##  1 2018-07-24 14:54:13 limited     4            74      6.97 2018-07-24
##  2 2018-07-24 14:54:13 some        8            27      7.74 2018-07-24
##  3 2018-07-24 14:54:13 none        3            26      6.81 2018-07-24
##  4 2018-07-24 14:54:13 limited     8            80      6.21 2018-07-24
##  5 2018-07-24 14:54:13 none        7            17      7.73 2018-07-24
##  6 2018-07-24 14:54:13 some       10            37      8.73 2018-07-24
##  7 2018-07-24 14:54:13 limited    10            46      6.67 2018-07-24
##  8 2018-07-24 14:54:13 lots        2            32      8.34 2018-07-24
##  9 2018-07-24 14:54:13 lots        3            55      7.71 2018-07-24
## 10 2018-07-24 14:54:13 lots       10            55      8.57 2018-07-24
## # ... with 40 more rows, and 1 more variable: Crisis <chr>
```

The `str()` or 'structure' command is also a great way to see what you've got.

```
str(survey)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    50 obs. of  7 variables:
##  $ Timestamp    : POSIXct, format: "2018-07-24 14:54:13" "2018-07-24 14:54:13" ...
##  $ R_exp        : chr  "limited" "some" "none" "limited" ...
##  $ Banjo        : int  4 8 3 8 7 10 10 2 3 10 ...
##  $ Psych_age_yrs: int  74 27 26 80 17 37 46 32 55 55 ...
##  $ Sleep_hrs    : num  6.97 7.74 6.81 6.21 7.73 ...
##  $ Fav_date     : Date, format: "2018-07-24" "2018-07-24" ...
##  $ Crisis       : chr  "Yes, significant" "Yes, slight" "Yes, slight" "Yes, significant" ...
##  - attr(*, "spec")=List of 2
##   ..$ cols   :List of 7
##   .. ..$ Timestamp    :List of 1
```

```
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_datetime" "collector"
##   .. ..$ R_exp        : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   .. ..$ Banjo        : list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Psych_age_yrs: list()
##   .. .. ..- attr(*, "class")= chr  "collector_integer" "collector"
##   .. ..$ Sleep_hrs    : list()
##   .. .. ..- attr(*, "class")= chr  "collector_double" "collector"
##   .. ..$ Fav_date     :List of 1
##   .. .. ..$ format: chr ""
##   .. .. ..- attr(*, "class")= chr  "collector_date" "collector"
##   .. ..$ Crisis       : list()
##   .. .. ..- attr(*, "class")= chr  "collector_character" "collector"
##   ..$ default: list()
##   .. ..- attr(*, "class")= chr  "collector_guess" "collector"
##   ..- attr(*, "class")= chr "col_spec"
```

Clearly, we need to do some cleaning before we can do anything with this.

Let's start by renaming variables.

```
names(survey) <- c("Timestamp",
                   "R_exp",
                   "Banjo",
                   "Psych_age_yrs",
                   "Sleep_hrs",
                   "Fav_day",
                   "Crisis")
```

```
# complete.cases() drops NAs
survey <- survey[complete.cases(survey),]
survey
```

```
## # A tibble: 50 x 7
##    Timestamp           R_exp   Banjo Psych_age_yrs Sleep_hrs Fav_day
##    <dttm>              <chr>   <int>         <int>     <dbl> <date>
##  1 2018-07-24 14:54:13 limited     4            74      6.97 2018-07-24
##  2 2018-07-24 14:54:13 some        8            27      7.74 2018-07-24
##  3 2018-07-24 14:54:13 none        3            26      6.81 2018-07-24
##  4 2018-07-24 14:54:13 limited     8            80      6.21 2018-07-24
##  5 2018-07-24 14:54:13 none        7            17      7.73 2018-07-24
##  6 2018-07-24 14:54:13 some       10            37      8.73 2018-07-24
##  7 2018-07-24 14:54:13 limited    10            46      6.67 2018-07-24
##  8 2018-07-24 14:54:13 lots        2            32      8.34 2018-07-24
##  9 2018-07-24 14:54:13 lots        3            55      7.71 2018-07-24
## 10 2018-07-24 14:54:13 lots       10            55      8.57 2018-07-24
## # ... with 40 more rows, and 1 more variable: Crisis <chr>
```

Now, lets make sure we have numbers where we expect them.

```
survey$Sleep_hrs <- readr::parse_number(survey$Sleep_hrs)
survey
```

```
## # A tibble: 50 x 7
##    Timestamp           R_exp   Banjo Psych_age_yrs Sleep_hrs Fav_day
```

```
##    <dttm>              <chr>   <int>         <int>     <dbl> <date>
##  1 2018-07-24 14:54:13 limited     4            74      6.97 2018-07-24
##  2 2018-07-24 14:54:13 some        8            27      7.74 2018-07-24
##  3 2018-07-24 14:54:13 none        3            26      6.81 2018-07-24
##  4 2018-07-24 14:54:13 limited     8            80      6.21 2018-07-24
##  5 2018-07-24 14:54:13 none        7            17      7.73 2018-07-24
##  6 2018-07-24 14:54:13 some       10            37      8.73 2018-07-24
##  7 2018-07-24 14:54:13 limited    10            46      6.67 2018-07-24
##  8 2018-07-24 14:54:13 lots        2            32      8.34 2018-07-24
##  9 2018-07-24 14:54:13 lots        3            55      7.71 2018-07-24
## 10 2018-07-24 14:54:13 lots       10            55      8.57 2018-07-24
## # ... with 40 more rows, and 1 more variable: Crisis <chr>
```

Looks good. Let's save that cleaned file so we don't have to do this again.

```
write_csv(survey, path="../data/survey_clean.csv")
```

We may want to make the `R_exp` variable ordered.

```
(survey_responses <- unique(survey$R_exp))
```

```
## [1] "limited" "some"    "none"    "lots"    "pro"
```

This shows us the different survey response values.

```
survey$R_exp <- ordered(survey$R_exp, levels=c("none",
                                               "limited",
                                               "some",
                                               "lots",
                                               "pro"))
```

Visualization

Now, we follow Mike Meyer's advice: "Plot your data!"

Descriptive plots

```
R_exp_hist <- survey %>%
  ggplot() +
  aes(x=R_exp) +
  geom_histogram(stat = "count") # R_exp is discrete
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
R_exp_hist
```

```
Sleep_hrs_hist <- survey %>%
  ggplot() +
  aes(x=Sleep_hrs) +
  geom_histogram() # Sleep_hrs is continuous
Sleep_hrs_hist
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Banjo_hist <- survey %>%
  ggplot() +
```

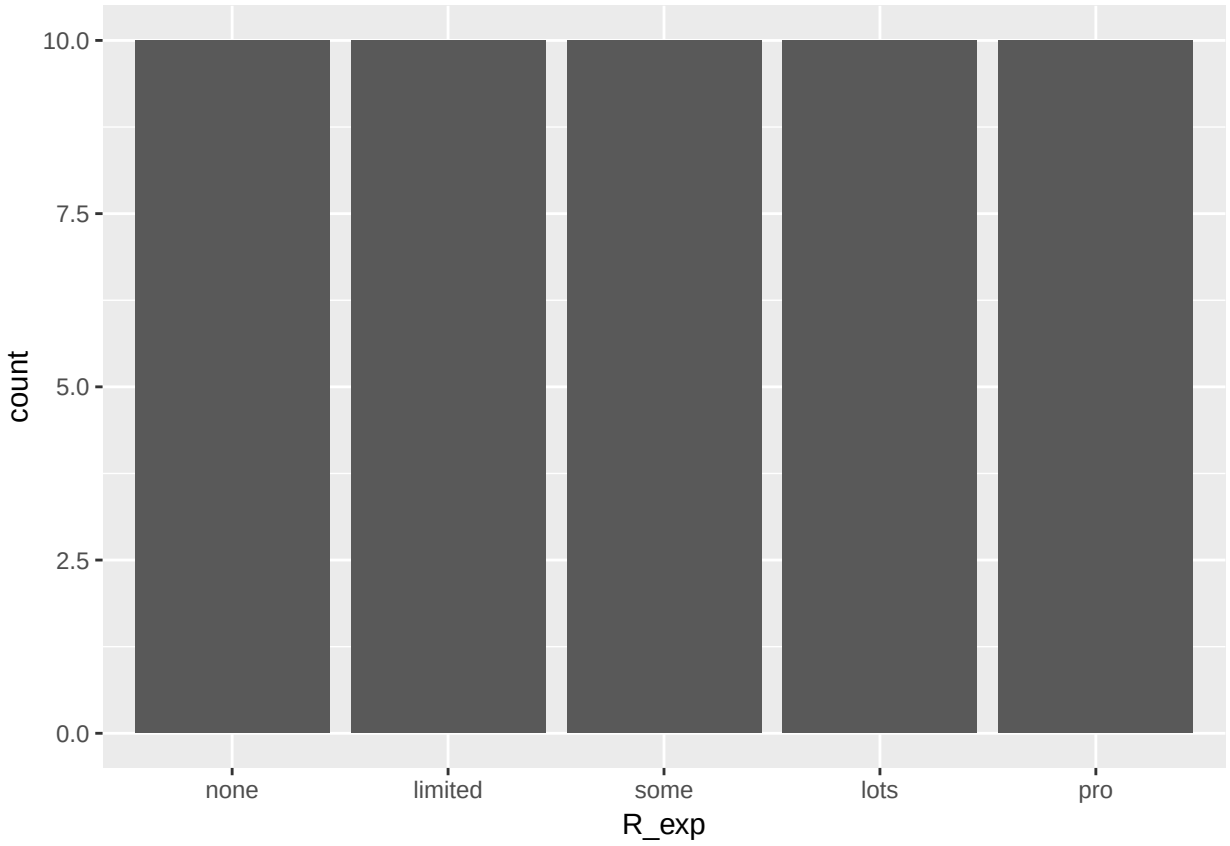


Figure 1: Distribution of prior R experience

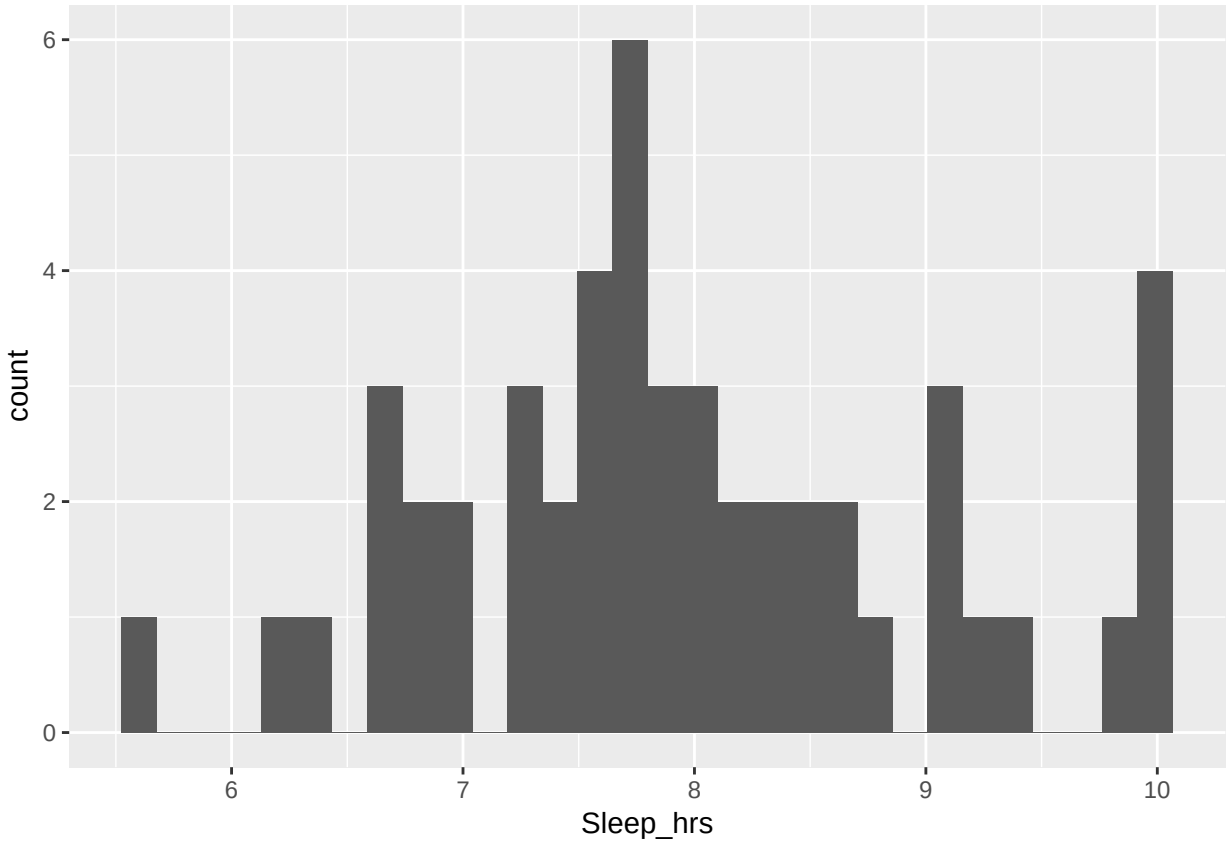


Figure 2: Distribution of preferred sleep hrs/day

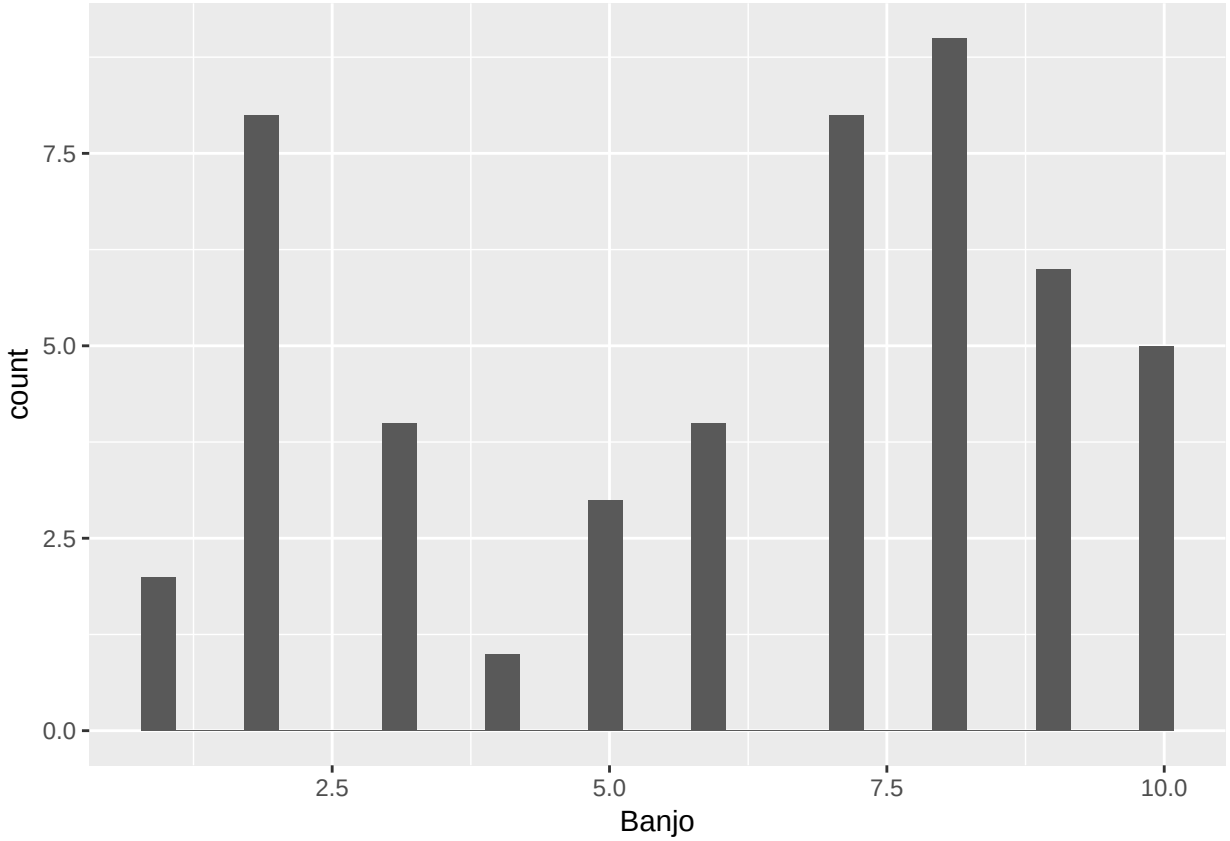


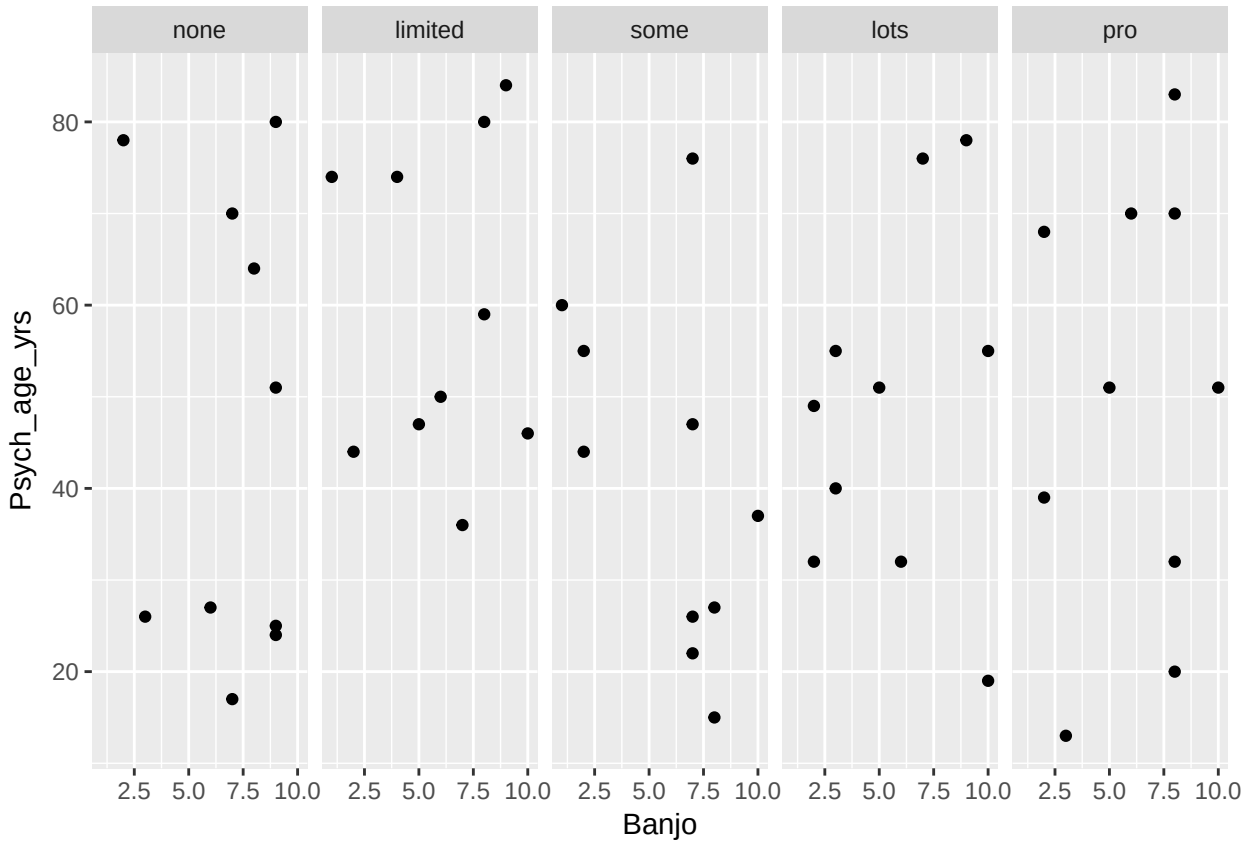
Figure 3: Distribution of Enthusiasm for Banjo Music

```
  aes(x=Banjo) +
  geom_histogram()
Banjo_hist
```

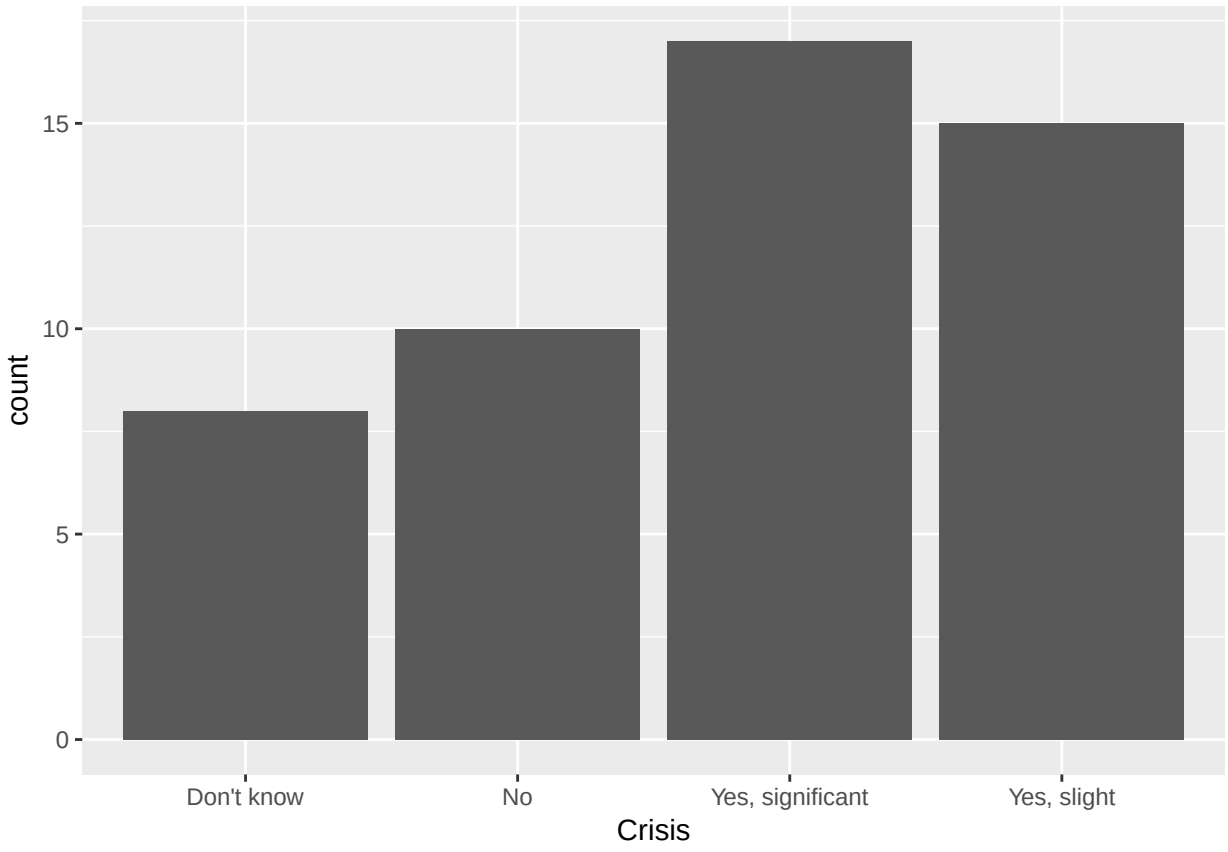
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Does R experience have any relation to banjo music enthusiasm?

```
Banjo_vs_r_exp <- survey %>%
  ggplot() +
  aes(x=Banjo, y=Psych_age_yrs) +
  facet_grid(. ~ R_exp) +
  geom_point()
  # + stat_smooth()
Banjo_vs_r_exp
```




```
crisis_plot <- survey %>%
  ggplot() +
  aes(x=Crisis) +
  geom_bar()
crisis_plot
```



Analysis

I could use a document like this to plan out my analysis plan **before** I conduct it. If I used simulated data, I could make sure that my workflow will run when I get real (cleaned) data. I could even preregister my analysis plan before I conduct it. That doesn't preclude later exploratory analyses, but it does hold me and my collaborators accountable for what I predicted in advance.

Notes

Notice that I sometimes put a label like `Banjo-vs-r-exp` in the brackets `{}`for a given 'chunk' of R code. The main reasons to do this are:

- It sometimes makes it easier to debug your code.
- In some cases, you can have this 'chunk' name serve as the file name for a figure you generate within a chunk.
- In a bit, we'll see how these chunk names are useful for making tables, figures, and equations that generate their own numbers.