

# Analysis of bootcamp survey

*Rick Gilmore*

*2017-08-16 18:07:21*

## Contents

Goals . . . . .	1
Preliminaries . . . . .	1
Load data and examine . . . . .	1
Visualization . . . . .	4
Descriptive plots . . . . .	4
Analysis . . . . .	7
Notes . . . . .	7

## Goals

- Download and clean data from 2017 R Bootcamp Survey
- Visualize data
- Prepare reports in `ioslides_presentation`, `pdf_document`, and `word_document` formats

## Preliminaries

Load required packages.

```
library(tidyverse)
library(googleheets)
```

## Load data and examine

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

There are some idiosyncrasies in using the `googleheets` 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, clean it, and save it to a CSV under `data/survey.csv`. We can then just load this file. But, let's look at `R/Clean_survey_data.R`.

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("../data/survey-test.csv")
# Or choose data from respondents
survey <- read_csv("../data/survey.csv")

## Parsed with column specification:
## cols(
##   Timestamp = col_character(),
```

```
## R_exp = col_character(),
## GoT = col_integer(),
## Age_yrs = col_integer(),
## Sleep_hrs = col_double(),
## Fav_day = col_character(),
## Tidy_data = col_character()
## )
```

```
survey
```

```
## # A tibble: 28 x 7
##       Timestamp    R_exp    GoT Age_yrs Sleep_hrs Fav_day
##       <chr>      <chr> <int>  <int>    <dbl>   <chr>
## 1 8/13/2017 23:29:24    some     10     28     8.0   Friday
## 2 8/14/2017 12:01:12    some     10     22     7.0   Friday
## 3 8/15/2017 12:42:09    some     10     24    10.0   Saturday
## 4 8/15/2017 17:13:08    none     10     28     9.0   Saturday
## 5 8/15/2017 19:03:40 limited     10     24     9.0   Saturday
## 6 8/15/2017 23:36:07    some     10     23     6.0   Friday
## 7 8/15/2017 23:45:05 limited      3     25     8.0   Friday
## 8 8/16/2017 0:26:01     pro      9     37     7.0   Friday
## 9 8/16/2017 1:09:44    none     10     25     9.0   Saturday
## 10 8/16/2017 8:51:05 limited      1     23     7.5   Thursday
## # ... with 18 more rows, and 1 more variables: Tidy_data <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': 28 obs. of 7 variables:
## $ Timestamp: chr "8/13/2017 23:29:24" "8/14/2017 12:01:12" "8/15/2017 12:42:09" "8/15/2017 17:13:08" ...
## $ R_exp : chr "some" "some" "some" "none" ...
## $ GoT : int 10 10 10 10 10 10 3 9 10 1 ...
## $ Age_yrs : int 28 22 24 28 24 23 25 37 25 23 ...
## $ Sleep_hrs: num 8 7 10 9 9 6 8 7 9 7.5 ...
## $ Fav_day : chr "Friday" "Friday" "Saturday" "Saturday" ...
## $ Tidy_data: chr "Yes" "That's a personal question" "No" "Yes" ...
## - attr(*, "spec")=List of 2
## ..$ cols :List of 7
## .. ..$ Timestamp: list()
## .. .. ..- attr(*, "class")= chr "collector_character" "collector"
## .. ..$ R_exp : list()
## .. .. ..- attr(*, "class")= chr "collector_character" "collector"
## .. ..$ GoT : list()
## .. .. ..- attr(*, "class")= chr "collector_integer" "collector"
## .. ..$ Age_yrs : list()
## .. .. ..- attr(*, "class")= chr "collector_integer" "collector"
## .. ..$ Sleep_hrs: list()
## .. .. ..- attr(*, "class")= chr "collector_double" "collector"
## .. ..$ Fav_day : list()
## .. .. ..- attr(*, "class")= chr "collector_character" "collector"
## .. ..$ Tidy_data: 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.

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

```
## # A tibble: 28 x 7
##       Timestamp    R_exp   GoT Age_yrs Sleep_hrs Fav_day
##       <chr>      <chr> <int>  <int>    <dbl>   <chr>
## 1 8/13/2017 23:29:24   some    10     28     8.0   Friday
## 2 8/14/2017 12:01:12   some    10     22     7.0   Friday
## 3 8/15/2017 12:42:09   some    10     24    10.0   Saturday
## 4 8/15/2017 17:13:08   none    10     28     9.0   Saturday
## 5 8/15/2017 19:03:40 limited    10     24     9.0   Saturday
## 6 8/15/2017 23:36:07   some    10     23     6.0   Friday
## 7 8/15/2017 23:45:05 limited     3     25     8.0   Friday
## 8 8/16/2017 0:26:01    pro     9     37     7.0   Friday
## 9 8/16/2017 1:09:44   none    10     25     9.0   Saturday
## 10 8/16/2017 8:51:05 limited     1     23     7.5   Thursday
## # ... with 18 more rows, and 1 more variables: Tidy_data <chr>
```

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

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

```
## # A tibble: 28 x 7
##       Timestamp    R_exp   GoT Age_yrs Sleep_hrs Fav_day
##       <chr>      <chr> <int>  <dbl>    <dbl>   <chr>
## 1 8/13/2017 23:29:24   some    10     28     8.0   Friday
## 2 8/14/2017 12:01:12   some    10     22     7.0   Friday
## 3 8/15/2017 12:42:09   some    10     24    10.0   Saturday
## 4 8/15/2017 17:13:08   none    10     28     9.0   Saturday
## 5 8/15/2017 19:03:40 limited    10     24     9.0   Saturday
## 6 8/15/2017 23:36:07   some    10     23     6.0   Friday
## 7 8/15/2017 23:45:05 limited     3     25     8.0   Friday
## 8 8/16/2017 0:26:01    pro     9     37     7.0   Friday
## 9 8/16/2017 1:09:44   none    10     25     9.0   Saturday
## 10 8/16/2017 8:51:05 limited     1     23     7.5   Thursday
## # ... with 18 more rows, and 1 more variables: Tidy_data <chr>
```

Looks good.

We may want to make the R\_exp variable ordered.

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

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

This shows us the different survey response values.

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

## Visualization

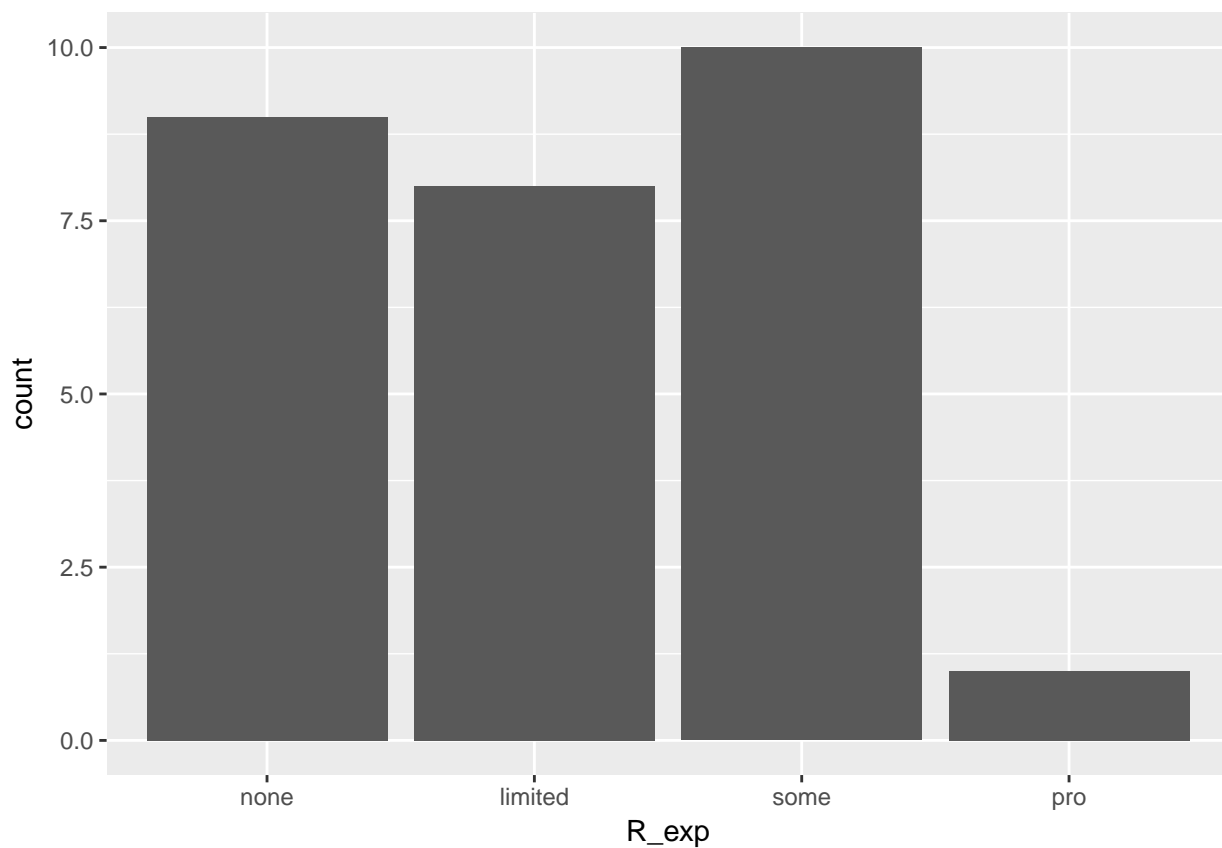
Now, we follow Mike Meyer's advice: "Plot your data!"

### Descriptive plots

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

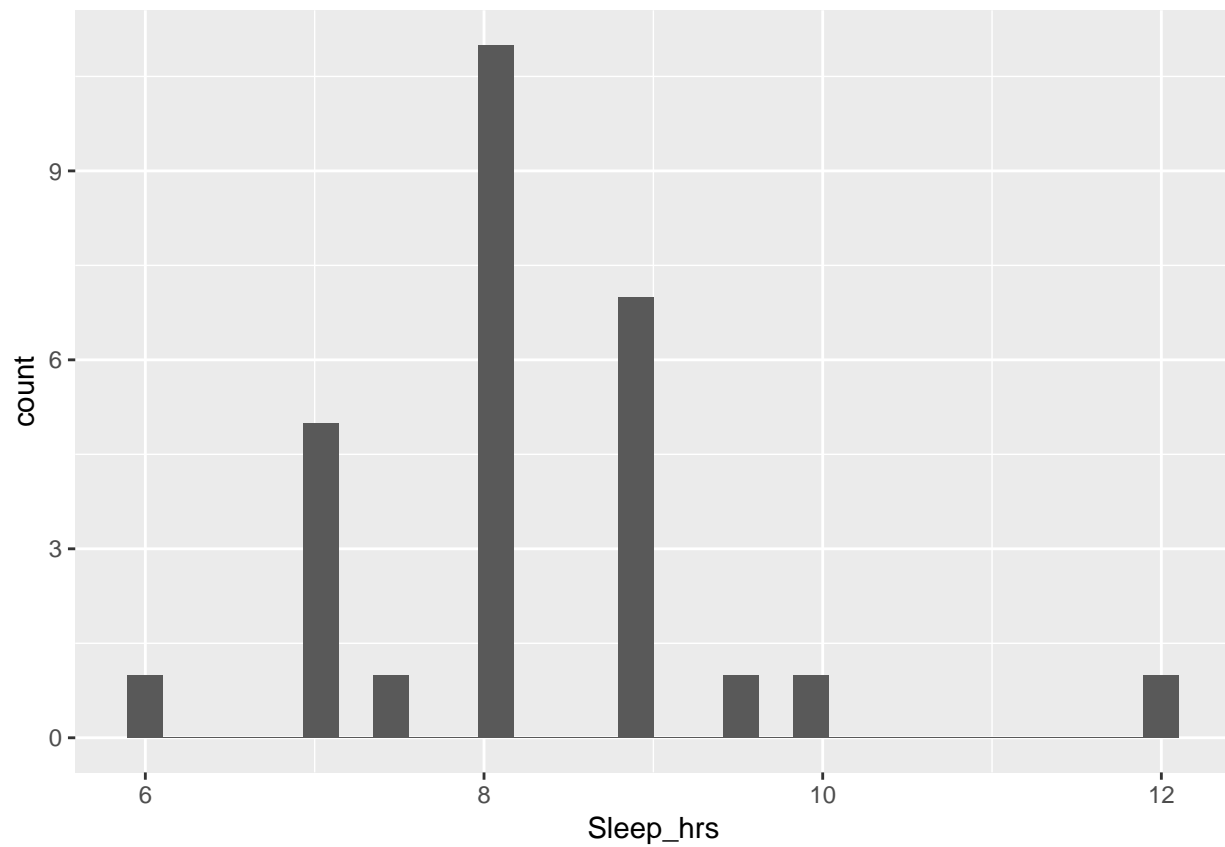
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
R_exp_hist
```



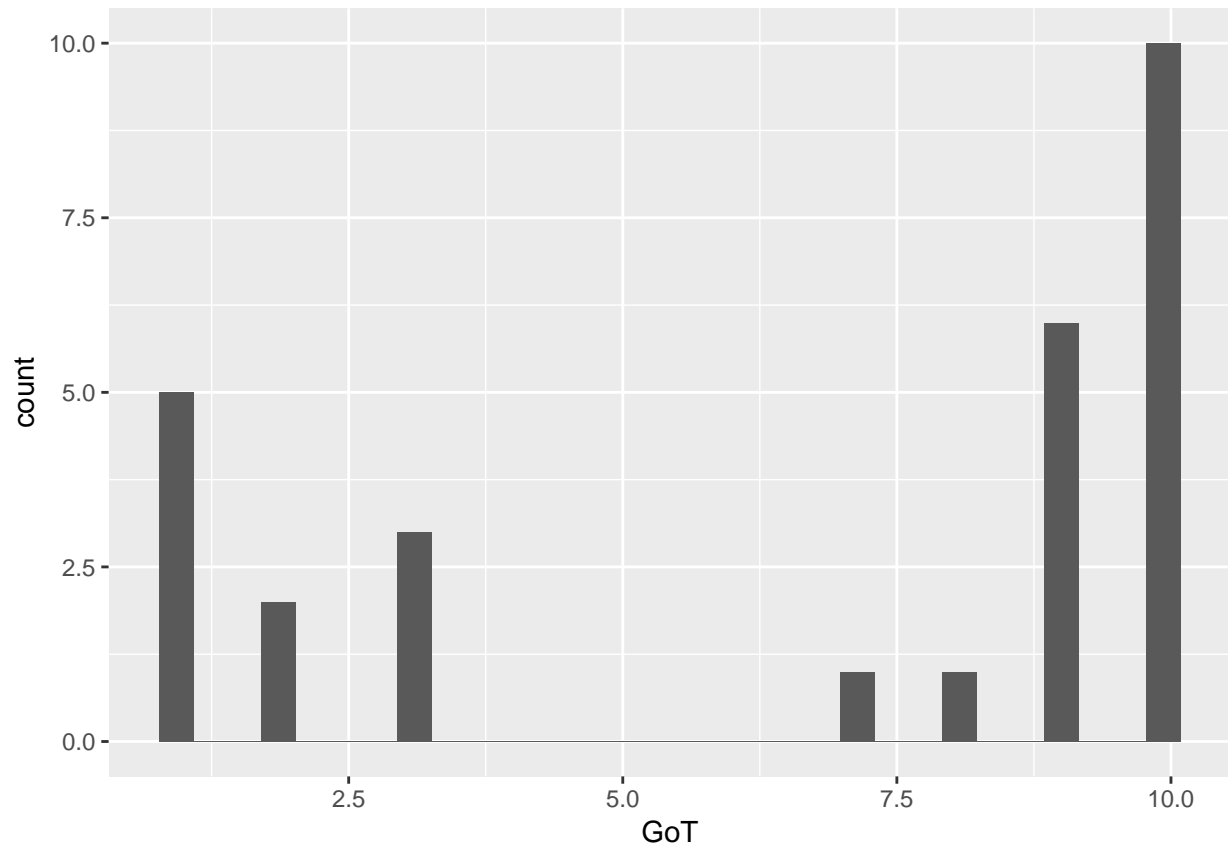
```
Sleep_hrs_hist <- survey %>%  
  ggplot() +  
  aes(x=Sleep_hrs) +  
  geom_histogram() # Sleep_hrs is continuous  
Sleep_hrs_hist
```

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



```
Got_hist <- survey %>%  
  ggplot() +  
  aes(x=GoT) +  
  geom_histogram()  
Got_hist
```

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

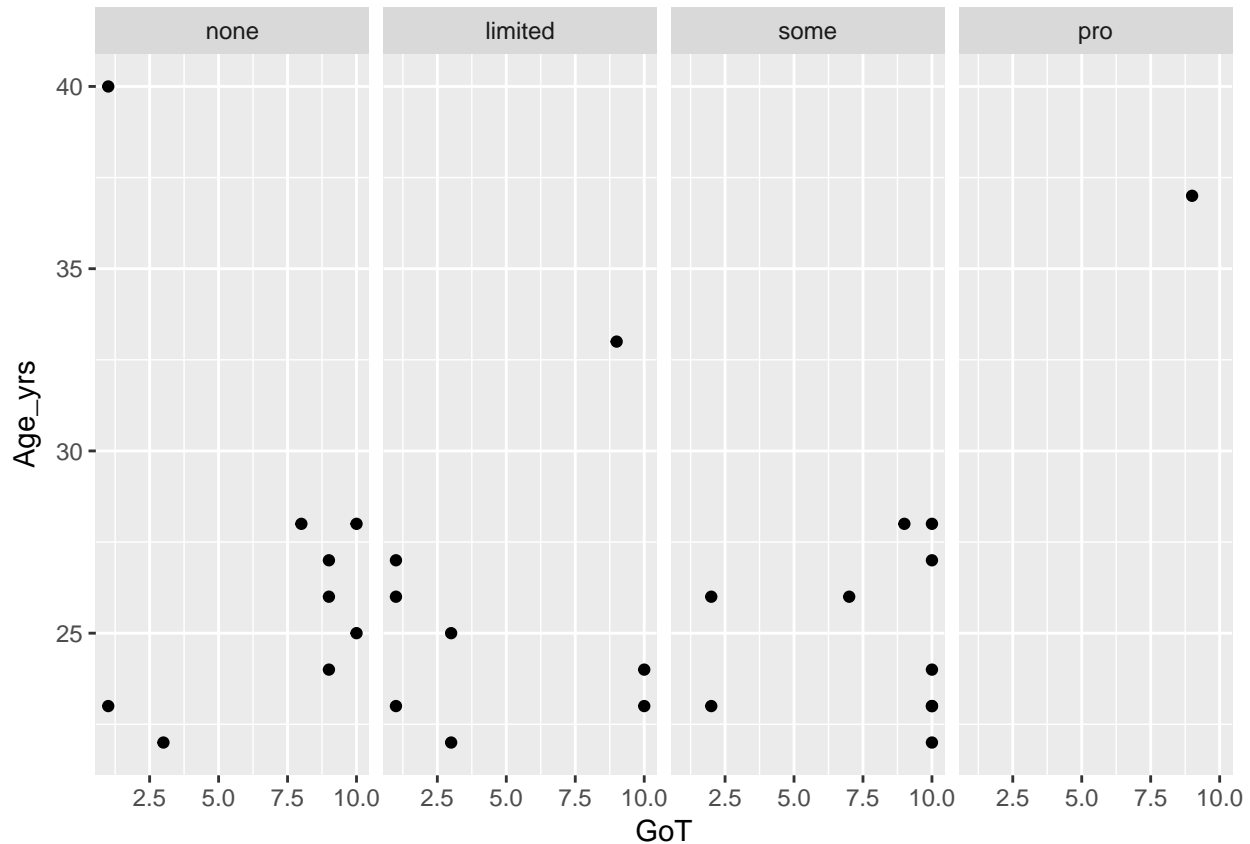


Looks like we are of two minds about GoT.



Figure 1:

```
GoT_vs_r_exp <- survey %>%
  ggplot() +
  aes(x=GoT, y=Age_yrs) +
  facet_grid(. ~ R_exp) +
  geom_point()
GoT_vs_r_exp
```



## 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 `got-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.