Analysis of bootcamp survey

Rick Gilmore

2017-08-17 08:50:22

Table of Contents

## 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(googlesheets)

## Load data and examine

The survey data are stored in a [Google Sheet](https://docs.google.com/spreadsheets/d/1Ay56u6g4jyEEdlmV2NHxTLBlcjI2gHavta-Ik0kGrpg/edit#gid=896447063). We'll use the googlesheets package to open it and create a data frame. Documentation about the package can be found [here](https://cran.r-project.org/web/packages/googlesheets/vignettes/basic-usage.html).

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, 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](../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](../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(),  
## `Your current level of experience/expertise with R` = col\_character(),  
## `Your enthusiasm for Game of Thrones` = col\_integer(),  
## `Age in years` = col\_integer(),  
## `Preferred number of hours spent sleeping/day` = col\_character(),  
## `Favorite day of the week` = col\_character(),  
## `Are your data tidy?` = col\_character()  
## )

survey

## # A tibble: 35 x 7  
## Timestamp `Your current level of experience/expertise with R`  
## <chr> <chr>  
## 1 <NA> <NA>  
## 2 8/13/2017 23:29:24 some  
## 3 8/14/2017 12:01:12 some  
## 4 8/15/2017 12:42:09 some  
## 5 8/15/2017 17:13:08 none  
## 6 8/15/2017 19:03:40 limited  
## 7 8/15/2017 23:36:07 some  
## 8 8/15/2017 23:45:05 limited  
## 9 8/16/2017 0:26:01 pro  
## 10 8/16/2017 1:09:44 none  
## # ... with 25 more rows, and 5 more variables: `Your enthusiasm for Game  
## # of Thrones` <int>, `Age in years` <int>, `Preferred number of hours  
## # spent sleeping/day` <chr>, `Favorite day of the week` <chr>, `Are your  
## # data tidy?` <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': 35 obs. of 7 variables:  
## $ Timestamp : chr NA "8/13/2017 23:29:24" "8/14/2017 12:01:12" "8/15/2017 12:42:09" ...  
## $ Your current level of experience/expertise with R: chr NA "some" "some" "some" ...  
## $ Your enthusiasm for Game of Thrones : int NA 10 10 10 10 10 10 3 9 10 ...  
## $ Age in years : int NA 28 22 24 28 24 23 25 37 25 ...  
## $ Preferred number of hours spent sleeping/day : chr NA "8!!!" "7" "10" ...  
## $ Favorite day of the week : chr NA "Friday" "Friday" "Saturday" ...  
## $ Are your data tidy? : chr NA "Yes" "That's a personal question" "No" ...  
## - attr(\*, "spec")=List of 2  
## ..$ cols :List of 7  
## .. ..$ Timestamp : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ Your current level of experience/expertise with R: list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ Your enthusiasm for Game of Thrones : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"  
## .. ..$ Age in years : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_integer" "collector"  
## .. ..$ Preferred number of hours spent sleeping/day : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ Favorite day of the week : list()  
## .. .. ..- attr(\*, "class")= chr "collector\_character" "collector"  
## .. ..$ Are your data tidy? : 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",  
 "GoT",  
 "Age\_yrs",  
 "Sleep\_hrs",  
 "Fav\_day",  
 "Tidy\_data")

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

## # A tibble: 34 x 7  
## Timestamp R\_exp GoT Age\_yrs Sleep\_hrs Fav\_day  
## <chr> <chr> <int> <int> <chr> <chr>  
## 1 8/13/2017 23:29:24 some 10 28 8!!! Friday  
## 2 8/14/2017 12:01:12 some 10 22 7 Friday  
## 3 8/15/2017 12:42:09 some 10 24 10 Saturday  
## 4 8/15/2017 17:13:08 none 10 28 9 Saturday  
## 5 8/15/2017 19:03:40 limited 10 24 9 Saturday  
## 6 8/15/2017 23:36:07 some 10 23 6-7 Friday  
## 7 8/15/2017 23:45:05 limited 3 25 8 Friday  
## 8 8/16/2017 0:26:01 pro 9 37 7 Friday  
## 9 8/16/2017 1:09:44 none 10 25 9 Saturday  
## 10 8/16/2017 8:51:05 limited 1 23 7.5 Thursday  
## # ... with 24 more rows, and 1 more variables: Tidy\_data <chr>

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

survey$Sleep\_hrs <- readr::parse\_number(survey$Sleep\_hrs)  
survey

## # A tibble: 34 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 24 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

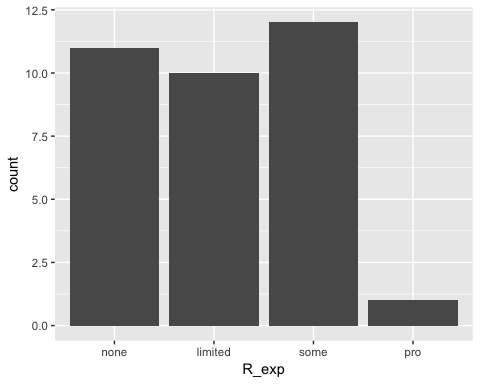
Now, we follow Mike Meyer's advice: "Plot your data!"

### Descriptive plots

R\_exp\_hist <- survey %>%  
 ggplot() +  
 aes(x=R\_exp) +  
 geom\_histogram(stat = "count") # R\_exp is discrete

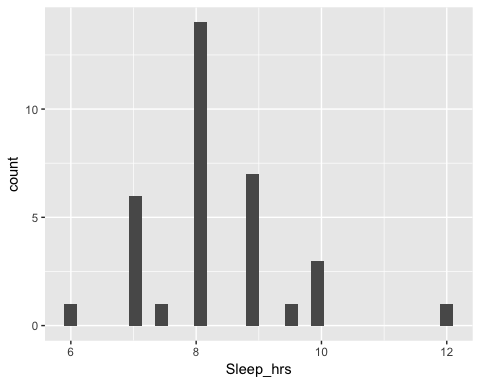
## Warning: Ignoring unknown parameters: binwidth, bins, pad

R\_exp\_hist



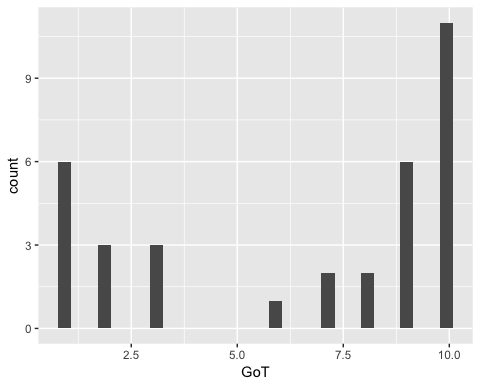
Sleep\_hrs\_hist <- survey %>%  
 ggplot() +  
 aes(x=Sleep\_hrs) +  
 geom\_histogram() # Sleep\_hrs is continuous  
Sleep\_hrs\_hist

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Got\_hist <- survey %>%  
 ggplot() +  
 aes(x=GoT) +  
 geom\_histogram()  
Got\_hist

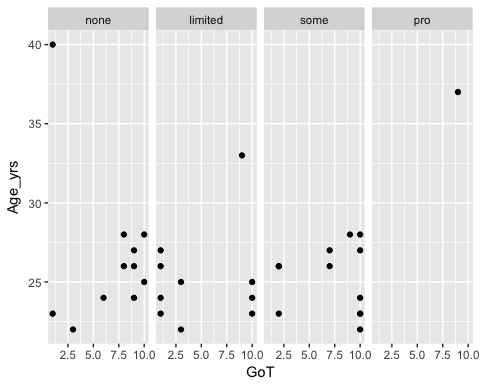
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Looks like we are of two minds about GoT.



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.