Interactive graphics

EDH7916 | Summer C 2020

Benjamin Skinner

In this supplemental lesson, we'll make a few interactive plots using the plotly library. The good news is that plotly works very similarly to ggplot. In fact, as you may have seen in a prior homework assignment, it's often trivial to convert a static ggplot figure into an interactive plotly figure.

Setup

```
## ---------------------------
## libraries
## ---------------------------

library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
✔ ggplot2 3.3.2          ✔ purrr   0.3.4
✔ tibble  3.0.3.9000     ✔ dplyr   1.0.1
✔ tidyr   1.1.0          ✔ stringr 1.4.0
✔ readr   1.3.1          ✔ forcats 0.5.0
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

```
library(haven)
library(plotly)
```

```
Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

As we did with the the lesson on making graphics, we'll use two sets of data: `hsls_small.dta` and `all_schools.csv`.

Note that since we have two data files this lesson, we'll give them unique names instead of the normal `df`:

- `df_hs` := `hsls_small.dta`

- df_ts := all_schools.csv

```
## ---------------------------
## directory paths
## ---------------------------

## assume we're running this script from the ./scripts subdirectory
dat_dir <- file.path("..", "data")
tsc_dir <- file.path(dat_dir, "sch_test")
```

```
## ---------------------------
## input data
## ---------------------------

## assume we're running this script from the ./scripts subdirectory
## read_dta() ==> read in Stata (*.dta) files
## read_csv() ==> read in comma separated value (*.csv) files
df_hs <- read_dta(file.path(dat_dir, "hsls_small.dta"))
df_ts <- read_csv(file.path(tsc_dir, "all_schools.csv"))
```

```
Parsed with column specification:
cols(
  school = col_character(),
  year = col_double(),
  math = col_double(),
  read = col_double(),
  science = col_double()
)
```

Plots using plotly

Histogram

```
## create histogram plotly
p <- plot_ly(data = df_hs, x = ~x1txmtscor, type = "histogram")
```

```
## show
p
```

```
## create histogram plotly
p <- plot_ly(data = df_hs,
             x = ~x1txmtscor,
             histnorm = "probability",
             type = "histogram")
```

```
## show
p
```

```
## create histogram plotly w/
## 1. better labels
## 2. add title
## 3. change color of bars
## 4. add outline to bars
p <- plot_ly(data = df_hs,
             x = ~x1txmtscor,
             histnorm = "probability",
```

```
            type = "histogram",
            marker = list(color = "#E28F41",
                          line = list(color = "#6C9AC3",
                                      width = 2))) %>%
    layout(title = "Distribution of math test scores",
           xaxis = list(title = "Math test score"))
```

```
## show
p
```

Plotting the difference in a continuous distribution across groups is a common task. Let's see the difference between student math scores between students with parents who have any postsecondary degree and those without.

Since we're using data that was labeled in Stata, we'll see the labels when we use `count()`

```
## see the counts for each group
df_hs %>% count(x1paredu)
```

```
# A tibble: 7 x 2
                                     x1paredu     n
                                    <dbl+lbl> <int>
1  1 [Less than high school]                   1010
2  2 [High school diploma or GED]              5909
3  3 [Associate's degree]                      2549
4  4 [Bachelor's degree]                       4102
5  5 [Master's degree]                         2116
6  7 [Ph.D/M.D/Law/other high lvl prof degree] 1096
7 NA                                           6721
```

```
## need to set up data
plot_df <- df_hs %>%
    ## select the columns we need
    select(x1paredu, x1txmtscor) %>%
    ## can't plot NA so will drop
    drop_na() %>%
    ## create new variable that == 1 if parents have any college
    mutate(pared_coll = ifelse(x1paredu >= 3, 1, 0)) %>%
    ## drop (using negative sign) the original variable we don't need now
    select(-x1paredu)

## show
head(plot_df)
```

```
# A tibble: 6 x 2
  x1txmtscor pared_coll
   <dbl+lbl>      <dbl>
1       59.4          1
2       47.7          1
3       64.2          1
4       49.3          1
5       62.6          1
6       58.1          1
```

```
## two way histogram
p <- plot_ly(alpha = 0.5) %>%
    add_histogram(data = plot_df %>% filter(pared_coll == 0),
```

```
                  x = ~x1txmtscor,
                  histnorm = "probability",
                  type = "histogram",
                  name = "No college") %>%
    add_histogram(data = plot_df %>% filter(pared_coll == 1),
                  x = ~x1txmtscor,
                  histnorm = "probability",
                  type = "histogram",
                  name = "Some college or more") %>%
    layout(barmode = "overlay",
           title = "Distribution of math test scores",
           xaxis = list(title = "Math test score"),
           legend = list(title = list(text = "Parent's education level")))
```

```
## show
p
```

Box plot

```
## box plot using both factor() and as_factor()
p <- plot_ly(data = df_hs,
             color = ~as_factor(x1paredu),
             y = ~x1txmtscor,
             type = "box") %>%
    layout(title = "Math score by parental expectations",
           yaxis = list(title ="Math score"),
           legend = list(title = list(text = "Parental expectations")))
```

```
## show
p
```

Scatter

```
## sample 10% to make figure clearer
df_hs_10 <- df_hs %>%
    ## drop observations with missing values for x1stuedexpct
    drop_na(x1stuedexpct) %>%
    ## sample
    sample_frac(0.1)
```

```
## scatter
p <- plot_ly(data = df_hs_10,
             x = ~x1ses,
             y = ~x1txmtscor,
             type = "scatter",
             mode = "markers")
```

```
## show
p
```

```
## scatter
p <- plot_ly(data = df_hs_10,
             x = ~x1ses,
             y = ~x1txmtscor,
```

```
            type = "scatter",
            mode = "markers",
            marker = list(size = 10,
                          color = "#E28F41",
                          line = list(color = "#6C9AC3",
                                      width = 2)),
            name = "",
            hovertemplate = paste("SES: %{x}",
                                  "<br>", # add <br> for line break
                                  "Math: %{y}")) %>%
    layout(title = "Math scores as function of SES",
           xaxis = list(title = "SES",
                        zeroline = FALSE),
           yaxis = list(title = "Math score"))
```

```
## show
p
```

```
## see student base year plans
df_hs %>%
    count(x1stuedexpct)
```

```
# A tibble: 12 x 2
                                   x1stuedexpct     n
                                      <dbl+lbl> <int>
 1  1 [Less than high school]                      93
 2  2 [High school diploma or GED]               2619
 3  3 [Start an Associate's degree]               140
 4  4 [Complete an Associate's degree]           1195
 5  5 [Start a Bachelor's degree]                 115
 6  6 [Complete a Bachelor's degree]             3505
 7  7 [Start a Master's degree]                   231
 8  8 [Complete a Master's degree]               4278
 9  9 [Start Ph.D/M.D/Law/other prof degree]      176
10 10 [Complete Ph.D/M.D/Law/other prof degree]  4461
11 11 [Don't know]                               4631
12 NA                                            2059
```

We see that `x1stuedexpct >= 6` means a student plans to earn a Bachelor's degree or higher. But since we need to account for the fact that 11 means "I don't know", we need to make sure our test includes `x1stuedexpct < 11`. Remember from a prior lesson that we can connect these two statements together with the operator `&`. Let's create our new variable.

```
## create variable for students who plan to graduate from college
df_hs_10 <- df_hs_10 %>%
    mutate(plan_col_grad = ifelse(x1stuedexpct >= 6 & x1stuedexpct < 11,
                                  "Yes",        # if T: 1
                                  "No"))       # if F: 0
```

Now that we have our new variable `plan_col_grad`, we can add it the `color` aesthetic, `aes()` in `geom_point()`. Don't forget to use `factor()` so that ggplot knows to treat it like a group!

```
## set color palette with names
pal <- c("#E28F41","#6C9AC3")
pal <- setNames(pal, c("Yes", "No"))
```

```r
## scatter
p <- plot_ly() %>%
    add_trace(data = df_hs_10,
              x = ~x1ses,
              y = ~x1txmtscor,
              color = ~plan_col_grad,
              colors = pal,             # using pal from above
              type = "scatter",
              mode = "markers",
              hovertemplate = paste("SES: %{x}",
                                    "<br>", # add <br> for line break
                                    "Math: %{y}")) %>%
    layout(title = "Math scores as function of SES",
           xaxis = list(title = "SES",
                        zeroline = FALSE),
           yaxis = list(title = "Math score"),
           legend = list(title = list(text = "Plans to graduate from college?")))
```

```r
## show
p
```

Line graph

```r
## show test score data
df_ts
```

```
# A tibble: 24 x 5
   school        year  math  read science
   <chr>        <dbl> <dbl> <dbl>   <dbl>
 1 Bend Gate     1980   515   281     808
 2 Bend Gate     1981   503   312     814
 3 Bend Gate     1982   514   316     816
 4 Bend Gate     1983   491   276     793
 5 Bend Gate     1984   502   310     788
 6 Bend Gate     1985   488   280     789
 7 East Heights  1980   501   318     782
 8 East Heights  1981   487   323     813
 9 East Heights  1982   496   294     818
10 East Heights  1983   497   306     795
# … with 14 more rows
```

```r
## reshape data long
df_ts_long <- df_ts %>%
    pivot_longer(cols = c("math","read","science"), # cols to pivot long
                 names_to = "test",                 # where col names go
                 values_to = "score") %>%           # where col values go
    group_by(test) %>%
    mutate(score_std = (score - mean(score)) / sd(score)) %>%
    group_by(test, school) %>%
    arrange(year) %>%
    mutate(score_year_one = first(score),
           ## note that we're using score_year_one instead of mean(score)
           score_std_sch = (score - score_year_one) / sd(score)) %>%
    ungroup
```

```
## show
df_ts_long
```

```
# A tibble: 72 x 7
   school        year test    score score_std score_year_one score_std_sch
   <chr>        <dbl> <chr>   <dbl>     <dbl>          <dbl>         <dbl>
 1 Bend Gate     1980 math      515     1.40             515             0
 2 Bend Gate     1980 read      281    -0.863            281             0
 3 Bend Gate     1980 science   808     0.759            808             0
 4 East Heights  1980 math      501     0.115            501             0
 5 East Heights  1980 read      318     1.34             318             0
 6 East Heights  1980 science   782    -0.735            782             0
 7 Niagara       1980 math      514     1.31             514             0
 8 Niagara       1980 read      292    -0.208            292             0
 9 Niagara       1980 science   787    -0.448            787             0
10 Spottsville   1980 math      498    -0.161            498             0
# … with 62 more rows
```

```
## scatter
p <- plot_ly() %>%
    add_trace(data = df_ts %>% filter(school == "Bend Gate"),
              x = ~year,
              y = ~math,
              name = "Math",
              type = "scatter",
              mode = "lines") %>%
    add_trace(x = ~year,
              y = ~read,
              name = "Reading",
              type = "scatter",
              mode = "lines") %>%
    add_trace(x = ~year,
              y = ~science,
              name = "Science",
              type = "scatter",
              mode = "lines") %>%
    layout(title = "Test scores at Bend Gate: 1980 - 1985",
           xaxis = list(title = "Year"),
           yaxis = list(title = "Score"),
           legend = list(title = list(text = "Test")),
           hovermode = "x unified")
```

```
## show
p
```

```
schools <- c("Bend Gate", "East Heights", "Niagara", "Spottsville")
plot_list <- list()
for (i in schools) {
    if_bg <- (i == schools[1])
    ## scatter
    plot_list[[i]] <- plot_ly() %>%
        add_trace(data = df_ts_long %>% filter(school == i),
                  x = ~year,
                  y = ~score_std_sch,
                  color = ~test,
```

```
                  legendgroup = ~test,
                  text = ~score,
                  showlegend = if_bg,
                  type = "scatter",
                  mode = "lines",
                  hovertemplate = paste("Year: %{x}",
                                        "<br>",
                                        "Score (scaled): %{y}",
                                        "<br>",
                                        "Score (actual): %{text}")) %>%
        layout(xaxis = list(title = "Year"),
               yaxis = list(title = "Score"),
               legend = list(title = list(text = "Test"))) %>%
        add_annotations(text = i,
                        x = 0,
                        y = 1,
                        yref = "paper",
                        xref = "paper",
                        xanchor = "middle",
                        yanchor = "top",
                        showarrow = FALSE,
                        font = list(size = 15))
}

p <- subplot(plot_list[[1]], plot_list[[2]], plot_list[[3]], plot_list[[4]],
             nrows = 2)
```

```
## show
p
```