Lab 6 Solutions

Steven Boyd

11/4/2021

PS5 Review

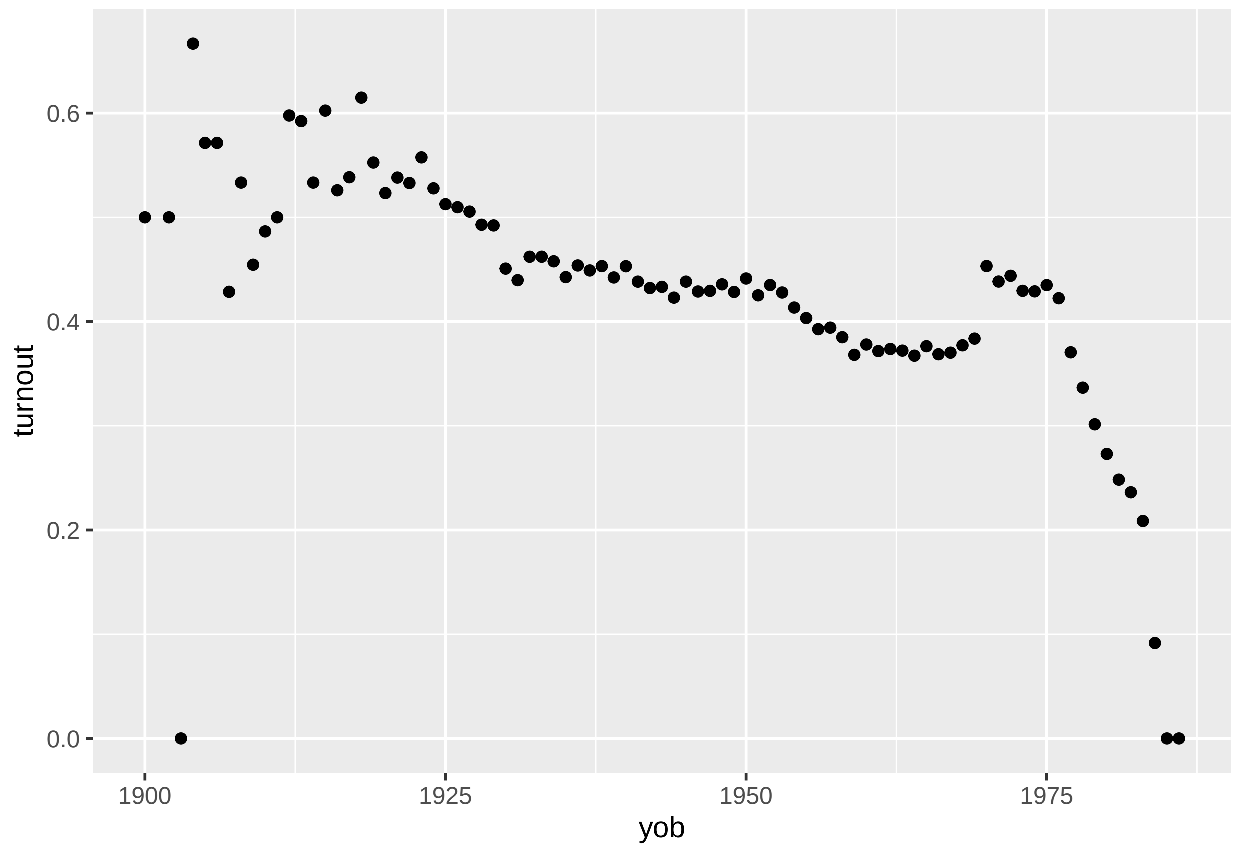
Question 1 asked you to compute the proportion of individuals who voted in the 2002 primaries by year of birth and then plot the proportions. Most people did this successfully:

```
load("data/ggl.RData")  #adjust filepath as necessary to load ggl.Rdata

ggl_prop <- ggl %>%
  group_by(yob) %>%
  summarize(turnout = mean(p2002, na.rm = TRUE))

plot_turnout <- ggplot(data = ggl_prop, aes(x = yob, y = turnout)) +
  geom_point()

plot_turnout
```



Questions 2, 3, and 4 asked you to run or plot a regression with voting in the 2002 primary as the dependent variable and year of birth as the independent variable. The most common mistake on the problem set was using the proportions calculated in question 1 as the dependent variable instead of `p2002` from `ggl`.

Let's see what happens to our estimated coefficient when we use these two different dependent variables:

```r
model_1 <- lm(p2002 ~ yob, data = ggl)

coef(model_1)
```

```
##  (Intercept)          yob
##  9.754675004 -0.004787226
```

```r
model_1_prop <- lm(turnout ~ yob, data = ggl_prop)

coef(model_1_prop)
```

```
##  (Intercept)          yob
##  6.656439206 -0.003205066
```

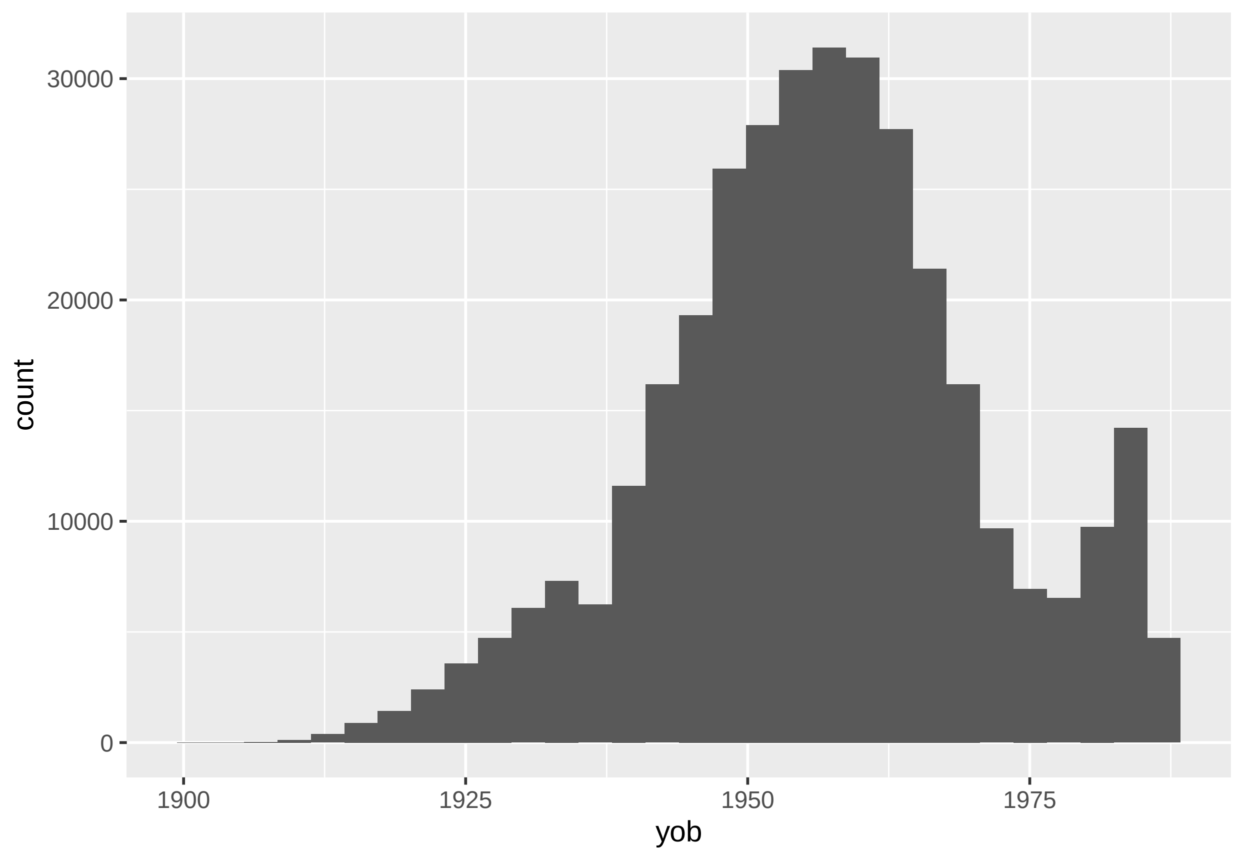
Notice the difference in the coefficient on `yob`? The difference may not seem that large in absolute terms (-.0032 vs. -.0048), but one is actually 50% larger than the other! When you recall that this coefficient is the predicted change in proportion of voters per 1 year increase in `yob`, you realize that the difference in the predicted turnout between the two models, especially at the ends, is quite large. Why?

The comment I left on many people's PS was that using the proportion as the DV in the regression was "overweighting" certain observations. Remember that OLS regression always minimizes the sum of square residuals.

When you regress onto the proportions, each proportion is given equal weight in the residual sum of squares calculation, even though there was not an equal number of observations in each year of birth. This is problematic because `yob` is not distributed uniformly:

```r
ggl %>%
  ggplot(aes(x = yob)) +
    geom_histogram()
```

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



When you regress onto the raw data (just a 0 or 1 for each observation), each of *those* points gets equal weight in the residual sum of squares calculation.

Summary tables and coefficient plots

In published quantitative work, you will often see *summary tables* and *coefficient plots*. These are different ways of presenting information from regressions and can be especially helpful when comparing between models. For example, it can help demonstrate that a statistically significant relationship in one model is *robust* to alternative specifications (or that someone else's result is not robust). Often, alternative specifications are created in anticipation of or response to feedback like: "you haven't controlled for *[insert possible confounder here]*."

When we talked about omitted variable bias, someone suggested that instead of speculating about the impact of variables left out of our model, we could simply include them in the model. So long as we have the right data, we can! Summary tables and coefficient plots give us a way to quickly compare between models with different combinations of independent variables (i.e. different "specifications"). First, let's look at an example of a summary table from Gartzke (2007), "The Capitalist Peace."

What information can you glean from it? What does each column represent? Why are some rows missing values in certain columns? Notice that the statistically significant coefficients from the first column are not significant in subsequent columns - what might we say about the apparent relationship from the first model?

We can see point estimates of the coefficients, standard errors (in parentheses), significance levels (indicated by `***`), the number of observations included in each model (N), and χ^2 for each model. Each column is a model specification and each row is a variable. A missing value indicates that the variable was omitted from that specification. Since the statistical significance on democracy indicators disappears in specifications that include economic variables, we could say that the significance of democracy is not robust to the inclusion of Gartzke's measures of capitalism.

Now, let's look at examples of coefficient plots in recently published papers. First, from the APSR: Alrababa'h et. al. (2021), "Can Exposure to Celebrities Reduce Prejudice? The Effect of Mohamed Salah on Islamophobic Behaviors and Attitudes."

In this plot, what do the labels on the left edge of the plot represent? What do the circles and bars represent?

This is a survey experiment, and each outcome corresponds to a binary survey question. The circle corresponds to the point estimate of the effect of the treatment on each outcome. The bars represent the confidence intervals.

Next, let's look at Eggers et. al. (2021), "No evidence for systematic voter fraud: A guide to statistical claims about the 2020 election," published in PNAS.

This visualization is a little bit different. What do the point shapes and colors represent? What do the labels on the left mean now? What do you think it means that the confidence interval does not include 0 in several of the Lott models, but does in the others?

The point shapes and colors correspond to variants of the model specification and the labels correspond to different models. The fact that 3 of the 4 Lott model confidence intervals not include 0 but the rest do indicates that Lott's claims about fraud in the 2020 US presidential election are not robust to small (and reasonable) changes to the data or model.

Creating summary tables and coefficient plots

It is possible to hand code tables and plots like this, but doing so can be extremely tedious. Luckily there are `R` packages which generate them for you.

We're going to focus on `modelsummary`. You can look at the documentation in R Studio, or use this link to access the vignette in your browser. After you've taken a moment to read through the documentation, let's create some models to practice (hopefully you are familiar with this data now).

```r
mod_1 <- lm(voted ~ treatment + sex, data = ggl)

mod_2 <- lm(voted ~ treatment + p2004, data = ggl)

mod_3 <- lm(voted ~ treatment + sex + p2004 + sex*p2004, data = ggl)

mods <- list(mod_1, mod_2, mod_3)
```

You don't necessarily need to save your list of models to a variable. In fact, you could generate models with `lm` within your call of `modelsummary` if you want to. Why might you want to save the list to a variable?

Saving the list of models as a variable allows us to try various settings in the table without having to type out the list of models we want to include every time. It saves time and reduces the risk of typos.

Let's see what the `modelsummary` default call looks like:

```r
modelsummary(mods)
```

Even with the default arguments, this is a pretty nice and informative table! Now, create another table, but this time save it to an html file called "mod_summary.html," add significance stars (`***`) to it, and exclude the intercept (hint: check the documentation). *This table will not render when you knit; it is being written to a file, so open the file to view it.*

```r
modelsummary(mods,
             output = "mod_summary.html",
             coef_omit = "Intercept",
             stars = TRUE)
```

```
## Warning: In version 0.8.0 of the `modelsummary` package, the default significance markers produced by
```

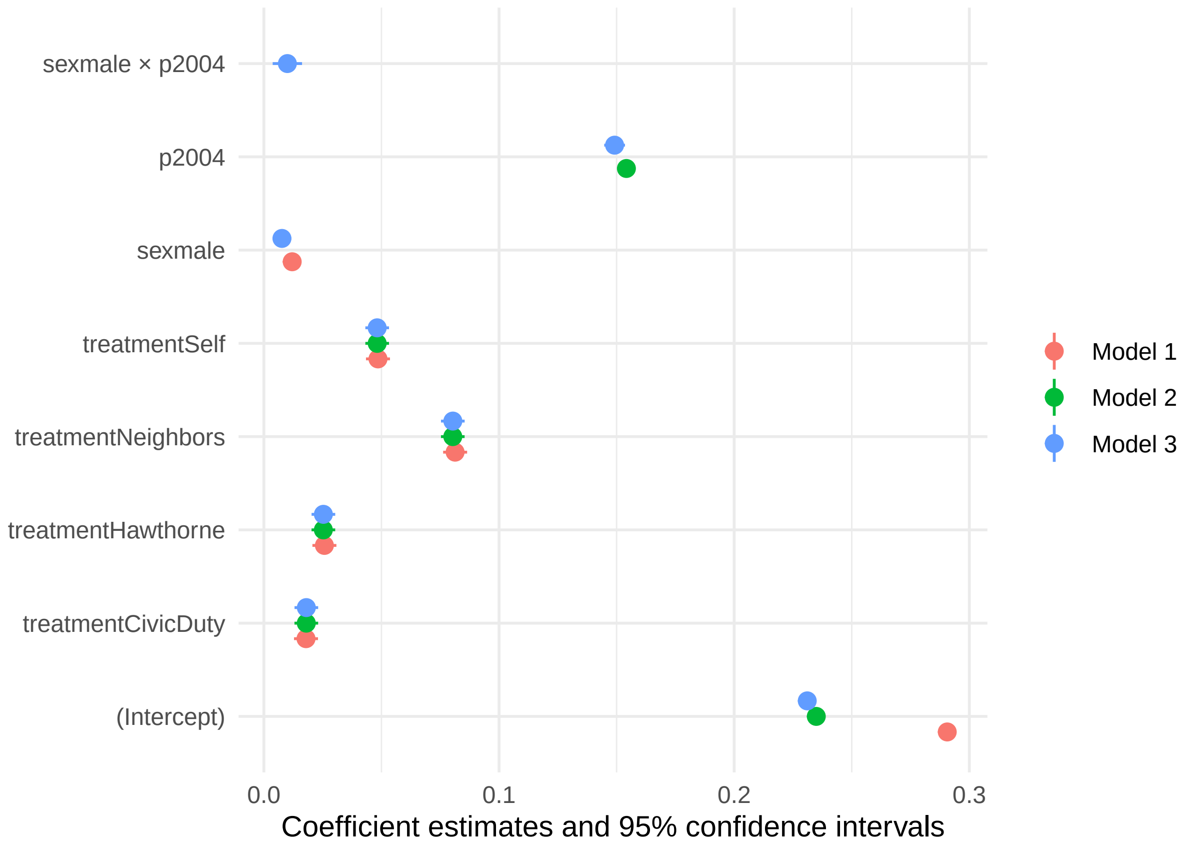
	Model 1	Model 2	Model 3
(Intercept)	0.291	0.235	0.231
	(0.001)	(0.001)	(0.002)
treatmentCivicDuty	0.018	0.018	0.018
	(0.003)	(0.003)	(0.003)
treatmentHawthorne	0.026	0.025	0.025
	(0.003)	(0.003)	(0.003)
treatmentNeighbors	0.081	0.080	0.080
	(0.003)	(0.003)	(0.003)
treatmentSelf	0.049	0.048	0.048
	(0.003)	(0.003)	(0.003)
sexmale	0.012		0.008
	(0.002)		(0.002)
p2004		0.154	0.149
		(0.002)	(0.002)
sexmale × p2004			0.010
			(0.003)
Num.Obs.	344 084	344 084	344 084
R2	0.004	0.030	0.030
R2 Adj.	0.004	0.030	0.030
AIC	448 124.0	438 933.6	438 871.6
BIC	448 199.3	439 008.8	438 968.3
Log.Lik.	−224 055.005	−219 459.776	−219 426.805
F	245.992	2115.478	1520.757

```
## This warning is displayed once per session.
```

Next, let's create a coefficient plot from the same models. The `modelsummary` package includes a function that does this too! Check the documentation for `modelplot` or read the vignette here.

Let's start with the default arguments again:

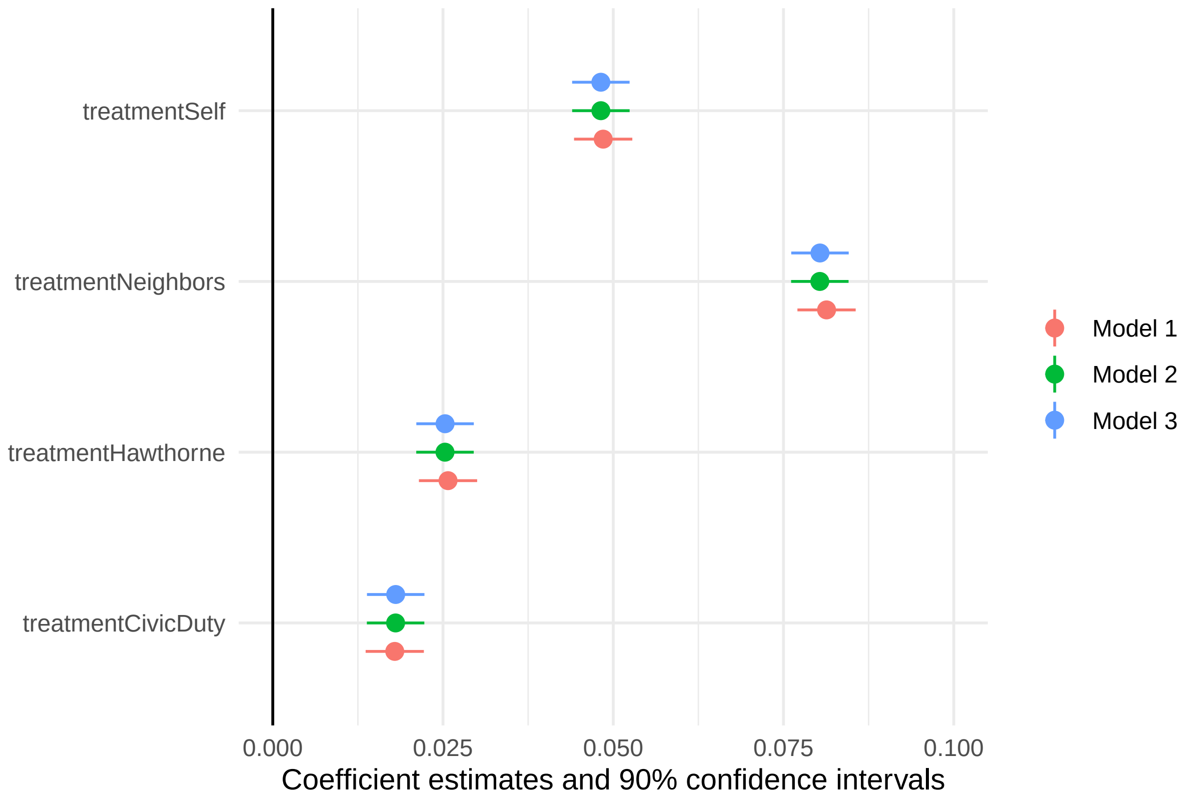
```
modelplot(mods)
```



This time, only show the coefficients on each treatment, display 90% confidence intervals, add a vertical line at the intercept, and title the plot. (hint: a nice feature of `modelsummary` is that it produces regular `ggplot2` objects, so you can add layers like we did when we discussed visualization)

```
modelplot(mods,
          conf_level = .9,
          coef_omit = "Intercept|sexmale|p2004"
          ) +
  geom_vline(xintercept = 0) +
  xlim(0, .1) +
  labs(title = "GGL Estimated Treatment Effects")
```

GGL Estimated Treatment Effects


treatmentSelf
treatmentNeighbors
treatmentHawthorne
treatmentCivicDuty
Coefficient estimates and 90% confidence intervals
Model 1
Model 2
Model 3