# Making the Most of Regression

POST 8000 – Foundations of Social Science Research for Public Policy

Steven V. Miller

Department of Political Science

# Goal for Today

*Discuss tricks to improve the information you can extract from a regression model.*

# Making the Most of Regression

Regression modeling is also storytelling.

- Your audience is not going to 100% understand what you're doing.
- That won't stop them from asking questions.

There's something you need to tell them from your model.

- *Make sure you tell them!*

Think of this like "Chekhov's gun."

- If it's in your model, be prepared to explain it.
- If you have to explain it, make it as intuitive as possible.

# Two Quickies

Here are two recurring things a lay audience will ask:

1. What is the “constant” or “y-intercept?”
2. Do bigger coefficients mean bigger effects relative to other coefficients?

# The Problem of the Constant

You know by now the "constant" or "y-intercept" is not a coefficient.

- It's just an estimate of $y$ when all $x$s are set to 0.

Your audience is going to want to interpret it.

- Worse yet: your audience is going to want to interpret something that probably makes no sense as you've been doing it.
- 0 is typically not an available or plausible response in raw regression inputs.

# What's a Bigger Effect?

Your coefficients will rarely, if ever, share a common scale.

- Absent a common scale, comparing regression coefficients is a fool's exercise.
- Coefficients are in part a function of the scale.
    - i.e. binary IVs will typically have larger coefficients (saying nothing of significance).

You and your audience will want to compare regression coefficients.

- However, your presentation will probably preclude this.

# A Simple Illustration

Let's illustrate this with a simple data set in {stevedata}.

```
election_turnout %>%
  select(state, percoled, gdp, sunempr12md, trumpshare)
```

```
## # A tibble: 51 x 5
##    state                percoled      gdp sunempr12md trumpshare
##    <chr>                   <dbl>    <dbl>       <dbl>      <dbl>
##  1 Alabama                  23.5  203830.        -0.2       0.621
##  2 Alaska                   28     49363.         0.3       0.513
##  3 Arizona                  27.5  311091         -0.600     0.487
##  4 Arkansas                 21.1  120375.        -0.6       0.606
##  5 California               31.4 2657798.        -0.300     0.316
##  6 Colorado                 38.1  329368.        -0.6       0.433
##  7 Connecticut              37.6  263696.        -0.700     0.409
##  8 Delaware                 30     69550.        -0.2       0.419
##  9 District of Columbia     54.6  129826.        -0.5       0.0407
## 10 Florida                  27.3  938774.        -0.400     0.490
## # ... with 41 more rows
```

# A Simple Illustration

Let's see if we can model the share of the vote Trump received in 2016 (`trumpshare`) with four variables:

- the percent of the state (25 and older) with a college diploma (`percoled`).
- the GDP of the state in 2016 (`gdp16`).
- Whether the state is in the South (`south`).
- The 12-month difference in the state unemployment rate for Nov. 2016 (`sunempr12md`).
    - Higher values = rising unemployment relative to point from Nov. 2015.

Note: this is clearly a simple exercise and we'll ignore some other model problems.

- e.g. DC is a huge outlier, which is why we'll ignore it.
- Probably should logarithmically transform the GDP variable too.

# The Statistical Model

```
election_turnout %>%
    mutate(south = ifelse(region == "South", 1, 0),
           # stored as a proportion, let's scale it to 100
           trumpshare = trumpshare*100) -> election_turnout


M1 <- lm(trumpshare ~ percoled + sunempr12md + gdp + south,
         subset(election_turnout, state != "District of Columbia"))
```

# The Output

```
broom::tidy(M1) %>%
    kable(.)
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | 96.4501468 | 7.0229871 | 13.7334934 | 0.0000000 |
| percoled | -1.5201759 | 0.2352240 | -6.4626732 | 0.0000001 |
| sunempr12md | 7.4415357 | 2.8906555 | 2.5743420 | 0.0134048 |
| gdp | -0.0000069 | 0.0000023 | -3.0400689 | 0.0039335 |
| south | 1.7090354 | 2.4006437 | 0.7119072 | 0.4801970 |

# Describing This Model

Here's how you would describe what you see here:

- A one-unit increase in % of the state with a college diploma decreases the Trump vote share by -1.52 percentage points.
- A one-unit increase in the state's GDP decreases Trump's vote share by about -.000007 points.
- A one-unit increase in the state unemployment change increases Trump's vote share by 7.44 points.
- Being in the South increases Trump's vote share by 1.7 points.

Takeaway: all but the South dummy had significant effects.

- However, there are several unsatisfying things about this model output.

# The Problem of the Constant

To start: the intercept suggests Trump's vote share is expected to be 96.45% (!) in a state where:

- No one graduated from college, AND
- There was no change from Nov. 2015 in the unemployment rate, AND
- The state isn't in the South, AND
- the GDP of the state is zero.

Constants/*y*-intercepts come standard in model output, but this parameter is useless as it is.

- You won't ever observe a case like this.

# The Problem of the Coefficients

What's the biggest effect, as a magnitude? You won't know.

- `percoled` is the most precise, but that's not magnitude.

All variables work on a different scale.

- `percoled` has a minimum of 19.2 (WV) and a maximum of 40.5 (MA).
- `gdp` has a minimum of 31,659 (VT) and a maximum of 2,657,798 (CA).
- `sunempr12md` has a minimum of -1 (NV) and a maximum of .4 (OH).
- `south` is a dummy/fixed effect and can only be 0 or 1.

# A Solution: Scaling (by Two Standard Deviations)

Statisticians recommend scaling your *non-binary* IVs, but Gelman (2008) has a better idea: **scale by two standard deviations instead of just one.**

- The transformed variable will have a mean of 0 and a SD of .5.
- Regression coefficient would communicate estimated change in $y$ for change across ~47.7% of data in $x$.

Sometimes this is what you want to communicate:

- i.e. do you really care about the effect of age going from 20 to 21? Or 50 to 51?
- Don't you want something larger/more substantive to communicate across the range of the data?

# The Added Benefit of Scaling by Two Standard Deviations

Gelman (2008) notes that scaling by 2 SDs instead of 1 puts non-binary IVs on a (roughly) common scale with binary IVs.

- Assume a dummy IV $d$ with a 50/50 split. Then: $p(d = 1) = .5$.
- Then, the SD equals .5 ($\sqrt{.5 * .5} = \sqrt{.25} = .5$)
- We can directly compare this dummy variable with our new standardized input variable!

This works well in most cases, except when $p(d = 1)$ is really small.

- e.g. $p(d = 1) = .25$, then $\sqrt{.25 * .75} = .4330127$

# How to Scale by Two Standard Deviations

The process looks similar to how to calculate a $z$-score (with the obvious change in the denominator).

- `rescale()` in the `{arm}` package will do it.
- `r2sd()` in my `{stevemisc}` package will do it.

You could also do it manually.

```
rescale <- function(x) { (x - mean(x, na.rm=T))/(2*sd(x, na.rm=T)) }
```

# How to Scale by Two Standard Deviations

```
election_turnout %>%
    # we'll use the custom function we just wrote in this document.
    mutate_at(vars("percoled", "gdp","sunempr12md"),
              list(z = ~rescale(.))) %>%
    rename_at(vars(contains("_z")),
              ~paste("z", gsub("_z", "", .), sep = "_") ) -> election_turnout

# Observe
mean(election_turnout$z_percoled)
```

```
## [1] -1.342641e-16
```

```
sd(election_turnout$z_percoled)
```

```
## [1] 0.5
```

# An Improved Statistical Model

```
M2 <- lm(trumpshare ~ z_percoled + z_sunempr12md + z_gdp +
             south, subset(election_turnout, state != "District of Columbia"))

broom::tidy(M2) %>%
    kable(.)
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | 47.186755 | 1.254211 | 37.6226496 | 0.0000000 |
| z_percoled | -18.417610 | 2.849844 | -6.4626732 | 0.0000001 |
| z_sunempr12md | 5.377487 | 2.088878 | 2.5743420 | 0.0134048 |
| z_gdp | -6.466175 | 2.126983 | -3.0400689 | 0.0039335 |
| south | 1.709035 | 2.400644 | 0.7119072 | 0.4801970 |

# Interpreting an Improved Statistical Model

- The typical state not in the South had an estimated Trump vote share of 47.18%. The intercept is much more informative.
  - Caveat: states are weighted equally, even as there are more Californians than South Carolinians.
  - i.e. you could've added Trump's SC votes to his CA tally and he'd still lose CA by >3 million votes.
- The education variable looks to have the largest effect when everything is on a mostly common scale.
  - i.e. the effect of going from, say, a standard deviation below the mean to a standard deviation above the mean is an estimated decrease in Trump's vote share by over 18 points.

# Interpreting an Improved Statistical Model

*Notice what didn't change.*

- Scaling the other variables doesn't change the binary IVs.
- The *t*-statistics don't change either even as the coefficients and standard errors change.

# Readable Regression Tables

Remember: your analysis should be as easily interpretable as possible.

- I should get a preliminary glimpse of effect size from a regression.
- Your $y$-intercept should be meaningful.

Standardizing variables helps.

- Creates meaningful zeroes (i.e. the mean).
- Coefficients communicate magnitude changes in $x$.
- Standardizing by two SDs allows for easy comparison with binary predictors.

# Satisfy Your Audience

You need to relate your analysis to both me and your grandma.

- I will obviously know/care more about technical details.
- Grandma may not, but she may be a more important audience than me.

Her inquiries will probably be understandable. Examples from the above analysis:

- What's Trump's expected vote share in a better-educated Southern state?
- What's Trump's expected vote share in a better-educated state whose unemployment rate increased?

These are perfectly reasonable questions to ask of your analysis.

- If your presentation isn't prepared to answer her questions, you're not doing your job.

# Statistical Presentations

Statistical presentations should:

1. Convey precise estimates of quantities of interest.
2. Include reasonable estimates of *uncertainty* around those estimates.
3. Require little specialized knowledge to understand Nos. 1 and 2.
4. Not bombard the audience with superfluous information.

We will do this with post-estimation simulation using draws from a multivariate normal distribution (King et al. 2000, Gelman and Hill, 2007).

# Estimating Uncertainty with Simulation

Any statistical model has a stochastic and systematic component.

- **Stochastic**: $Y_i \sim f(y_i \mid \theta_i, \alpha)$
- **Systematic**: $\theta_i = g(x_i, \beta)$

For a simple OLS model (i.e. a linear regression):

$$\begin{aligned} Y_i &= N(\mu_i, \sigma^2) \\ \mu_i &= X_i\beta \end{aligned}$$

# Understanding our Uncertainty

We have two types of uncertainty.

1. **Estimation uncertainty**
    - Represents systematic components; can be reduced by increasing sample size.
2. **Fundamental uncertainty**
    - Represents stochastic component; exists no matter what (but can be modeled).

# Getting our Parameter Vector

We want a **simulated parameter vector**, denoted as:

$$\hat{\gamma} \sim vec(\hat{\beta}, \hat{\alpha})$$

Central limit theorem says with a large enough sample and bounded variance:

$$\tilde{\gamma} \sim N(\hat{\gamma}, \hat{V}(\hat{\gamma}))$$

In other words: distribution of quantities of interest will follow a multivariate normal distribution with mean equal to $\hat{\gamma}$, the simulated parameter vector.

- Practically, whenever estimating a regression model, make sure you have your coefficients and variance-covariance matrix handy.

# Getting our Quantities of Interest

This is a mouthful! Let's break the process down step-by-step.

1. Run your regression. Get your results.
2. Choose values of explanatory variable (as you see fit).
3. Obtain simulated parameter vector from estimating systematic component.
4. Simulate the outcome by taking random draw from the stochastic component.

Do this $m$ times (typically $m = 1000$) to estimate the full probability distribution of your quantity of interest.

# How Do You Do This?

There are a variety of packages that can do this:

- `{Zelig}` is primarily responsible for this movement.
- `sim()` in the `{arm}` package can make simulations from a multivariate normal distribution.
- `{tidybayes}` is a go-to for Bayesian approaches (which give you these for free anyway).
    - That's next week, though.

My approach leans on `arm::sim()`.

- This is the foundation for my `get_sims()` function in `{stevemisc}`.

# A Quantity of Interest

What's Trump's expected vote share in:

1. a better-educated state where the unemployment rate got better over 12 months?
2. a better-educated state where the unemployment rate got worse over 12 months?
3. a lesser-educated state where the unemployment rate got better over 12 months?
4. a lesser-educated state where the unemployment rate got worse over 12 months?

Here's how to approach doing this.

- Caveat: this is obviously a low-powered cross-sectional, observational data set of a prominent event where proper nouns matter.
- But: the approach is still informative for a wide variety of applications.

# Create a New Data Frame

First, create a new data frame that match these parameters.

```
data_grid(.model = M2, data=election_turnout,
          trumpshare = 0,
          z_percoled = c(-.5, .5),
          z_sunempr12md = c(-.5, .5)) -> newdat
```

# Create a New Data Frame

```
newdat
```

```
## # A tibble: 4 x 5
##   trumpshare z_percoled z_sunempr12md  z_gdp south
##        <dbl>      <dbl>         <dbl>  <dbl> <dbl>
## 1          0       -0.5          -0.5 -0.163     0
## 2          0       -0.5           0.5 -0.163     0
## 3          0        0.5          -0.5 -0.163     0
## 4          0        0.5           0.5 -0.163     0
```

# Create a Model Matrix from the Model and this new data.frame

```
MM <- model.matrix(terms(M2),newdat)
```

# Get Simulations of Model Parameters via arm::sim()

```
set.seed(8675309) # Jenny, I got your number
simM2 <- arm::sim(M2, n.sims = 1000)
# ^ will grab your coefficients and vcov and do all that for you
```

# Here's a Glimpse of These Simulated Betas

```
as_tibble(coef(simM2)[1:5,])
```

```
## # A tibble: 5 x 5
##   `(Intercept)` z_percoled z_sunempr12md z_gdp south
##           <dbl>      <dbl>         <dbl> <dbl> <dbl>
## 1          47.1      -21.9          6.31 -5.19  1.86
## 2          48.6      -18.5          2.29 -8.34 -1.46
## 3          46.6      -21.5          4.33 -4.11  1.45
## 4          49.5      -20.8          1.83 -7.45 -2.83
## 5          48.2      -20.6          5.10 -6.41  1.92
```

# Calculate/Store Quantities of Interest

```
# Create blank Sims object
Sims <- tibble(y = numeric(),
               sim = numeric())

# For the 1,000 sims we have...
for(i in (1:1000)) {
  hold_me <- NULL # blank hold_me object
  # matrix multiplication from our model matrix with simulated coefs
  yi <- MM %*% coef(simM2)[i,]
  # note which of the 1,000 simulations it is.
  sim <-rep(i, length (yi))
  # cbind the QIs with the simulation indicator
  hold_me <- as_tibble(cbind(yi, sim)) %>% rename(y = V1) #
  # Bind the simulations together...
  Sims <- bind_rows(Sims, hold_me)
}
```

# Calculate/Store Quantities of Interest

```
Sims
```

```
## # A tibble: 4,000 x 2
##        y   sim
##    <dbl> <dbl>
##  1  55.7     1
##  2  62.0     1
##  3  33.8     1
##  4  40.2     1
##  5  58.1     2
##  6  60.4     2
##  7  39.6     2
##  8  41.9     2
##  9  55.9     3
## 10  60.2     3
## # ... with 3,990 more rows
```

# Remember What You're Looking At

Next, take inventory of what you're looking at based on the `newdat` object.

```
newdat %>%
    # replicate newdat 1000 times for our 1000 sims
    slice(rep(row_number(), 1000)) %>%
    # bind_col
    bind_cols(Sims, .) -> Sims
```

# Summarize As You See Fit

The world is your oyster when you do these post-estimation simulations.

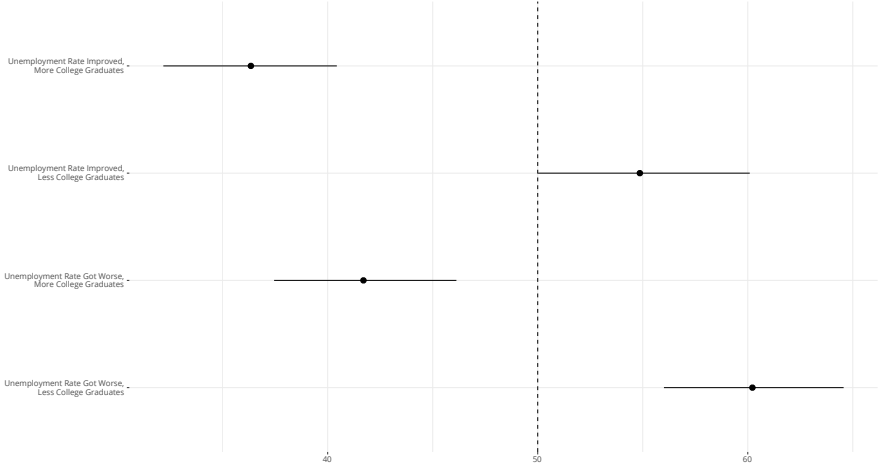
```
Sims %>%
    group_by(z_sunempr12md, z_percoled) %>%
    summarize(mean = mean(y),
              lwr = quantile(y, .025),
              upr = quantile(y, .975))
```

```
## `summarise()` has grouped output by 'z_sunempr12md'. You can overrid
```

```
## # A tibble: 4 x 5
## # Groups:   z_sunempr12md [2]
##   z_sunempr12md z_percoled  mean   lwr   upr
##           <dbl>      <dbl> <dbl> <dbl> <dbl>
## 1          -0.5       -0.5  54.9  50.0  60.1
## 2          -0.5        0.5  36.3  32.2  40.4
## 3           0.5       -0.5  60.2  56.0  64.6
## 4           0.5        0.5  41.7  37.4  46.1
```

## The Simulated Effect of Education Levels and Unemployment Rate Changes in the 2016 Presidential Election

The effect of education levels overpowers all other model inputs and is enough to effectively 'offset' an improving economy in the 2016 election.



# Conclusion

Regression provides all-else-equal effect sizes across the range of the data.

- You can extract meaningful quantities of interest from regression output itself.
- Typically, you'll need more to answer substantive questions and provide meaningful quantities of interest.
- You can help yourself by scaling your non-binary inputs by two SDs.

Post-estimation simulation from a multivariate normal distribution does this.

- When you start doing this yourselves, be prepared to provide quantities of interest for your audience.
- Never forget: *you're trying to tell a story.* Tell it well.

# Table of Contents