Making the Most of Regression (“Divide By 4”, Scaling)

POSC 3410 – Quantitative Methods in Political Science

Steven V. Miller

Department of Political Science

Goal for Today

Make the most of regression by making coefficients directly interpretable.

Introduction

You all should be familiar with regression by now.

Introduction

Regression coefficients communicate:

- Estimated change in y for one-unit change in x.
    - This is in linear regression.
- Estimated change in *logged odds* of y for one-unit change in x.
    - This is the interpretation for logistic regression.

These communicate some quantities of interest.

- After all, you want to know the effect of x on y!

Introduction

However, it's easy (and tempting) to provide misleading quantities of interest.

- Our variables are seldom (if ever) on the same scale.
    - e.g. age can be anywhere from 18 to 100+, but years of education are typically bound between 0 and 25 (or so).
- Worse yet, zero may not occur in any variable.
    - We would have an uninterpretable y-intercept.
    - From my experience, this can lead to false convergence of the model itself.

Your goal: regression results should be as easily interpretable as possible.

- Today will be about how to do that.

R Code/Packages for Today

```r
library(tidyverse) # for most things
library(stevemisc) # for formatting and r2sd()
library(stevedata) # for ?TV16
library(modelsummary) # for tables
library(kableExtra) # for prettying up tables

TV16 %>%
  filter(state == "Pennsylvania" & racef == "White") -> Penn
```

Gelman's Parlor Tricks

Andrew Gelman (2006 [with Hill], 2008) has two parlor tricks for getting the most out of regression.

1. The "divide by 4" rule for logistic regression coefficients.
2. Scaling by two standard deviations instead of one.

The "Divide by 4" Rule

OLS coefficients are intuitive.

- One unit increase in x increases estimated value of y.

Logistic regression coefficients are not intuitive (yet).

- One unit increase in x increases estimated natural logged odds of y.

The "Divide by 4" Rule

Gelman and Hill (2006, 82) argue you can extract more information from your coefficient if you know about the logistic curve.

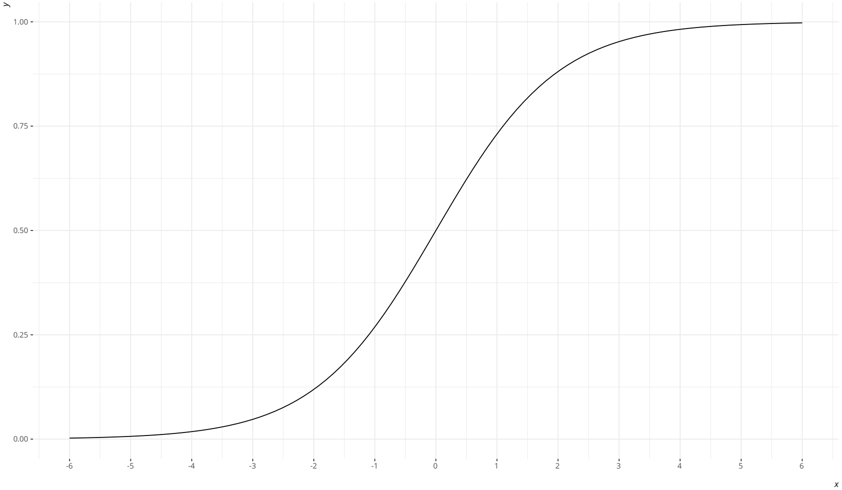
- The logistic curve is a familiar "S-curve" that transforms continuous variables to range from 0 to 1.

It'll look something like this.

```
tibble(x = seq(-6, 6)) %>%
  ggplot(.,aes(x)) +
  stat_function(fun =  function(x) exp(x)/(1+exp(x)))
```

The Logistic Curve

Notice the bounds between 0 and 1 on the y-axis for an unbounded x-axis. Also notice the curve is steepest in the middle.



The "Divide by 4" Rule

See how the curve is steepest in the middle? Remember derivatives from calc?

- It means that's the point where the slope is maximized.

That means it attains the value where

$$\beta e^0/(1+e^0)^2 = \beta/(1+1)^2 = \beta/4$$

Dividing a logistic regression coefficient by 4 gives you a reasonable *upper bound* of the predictive difference in *y* for a unit difference in *x*.

An Example

Let's assume we want to explain the white Trump vote in PA in 2016 as a function of education.

- *y*: respondent voted for Trump (Y/N)
- *x*: respondent has a four-year college diploma (Y/N)

```
M1 <- glm(votetrump ~ collegeed, data=Penn,
          family=binomial(link="logit"))

tidyM1 <- broom::tidy(M1)
interceptM1 <- tidyM1[1, 2] %>% pull()
coefM1 <- tidyM1[2, 2] %>% pull()
```

Table 1: Predicting the White Trump Vote in 2016 (CCES, 2016)

	Did White PA Respondent Vote for Trump?
College Educated	-0.818***
	(0.093)
Intercept	0.370***
	(0.055)
Num.Obs.	2124

* p < 0.1, ** p < 0.05, *** p < 0.01

An Example

Interpretation here is straightforward, but not too intuitive.

- The natural logged odds of voting for Trump for those without college education is 0.37.
- College education decreases those natural logged odds by -0.818.

Divide that coefficient by 4 and you get -0.205.

- That's an upper bound of the estimated effect in the probability of a white vote for Trump in PA for having a college diploma.

“Divide by 4” vs. DIY

It’s actually a really good heuristic!

```
# Gelman's divide by 4
coefM1/4
```

```
## [1] -0.20453
```

```
# Manually estimating the difference from the regression
plogis((interceptM1 + coefM1)) - plogis(interceptM1)
```

```
## [1] -0.2016522
```

Where *p(y = 1)* isn’t too small or large, this will do quite well when you look at your logistic regression output.

Standardize (by Two Standard Deviations)

Multiple regression models will have some other difficulties.

- Predictors will include variables on different scales (e.g. age in years, or male-female gender).
- Intercepts will come in tow, but may not make sense.

Variables will almost never share the same scale.

- Thus, you can't compare coefficients to each other, only to a null hypothesis of zero effect.

Standardize (by Two Standard Deviations)

Gelman (2008) offers a technique for interpreting regression results: scale the non-binary input data by two standard deviations.

- This makes continuous inputs (roughly) on same scale as binary inputs.
- It allows a preliminary evaluation of relative effect of predictors otherwise on different scales.

Why Two Instead of One?

Scaling by one standard deviation has important benefits.

- Scale variable has mean of 0 and standard deviation of 1.
- Communicates magnitude change across 34% of the data.
- Creates meaningful *y*-intercept (that approximates a mean/typical case).
- However, it won't help us make preliminary comparisons with dummy variables.

Scaling by two standard deviations has more benefits.

- Scale variable has mean of 0 and standard deviation of .5.
- Creates magnitude change across 47.7% of the data.
- Puts continuous inputs on roughly same scale as binary inputs.

How Does This Work?

Consider a dummy IV with 50/50 split between 0s and 1s.

- $p(dummy = 1) = .5$
- Then, standard deviation 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 $p(dummy = 1)$ is really small.

- e.g. $p(dummy = 1) = .25$, then $\sqrt{.25 * .75} = .43$

An Extended Example

Let's go back to our white Pennsylvanian data.

- *DV*: did respondent vote for Trump? (Y/N)
- *IV*s: age [18:88], gender (female), college education, household income [1:12], L-C ideology [1:5], D-R partisanship [1:7], respondent is born-again Christian.

```
M2 <- glm(votetrump ~ age + female + collegeed + famincr + ideo +
             pid7na + bornagain, data=Penn,
          family=binomial(link="logit"))

tidyM2 <- broom::tidy(M2)
```

Table 2: Predicting the White Trump Vote in 2016 (CCES, 2016)

	Did White PA Respondent Vote for Trump?
Age	0.010**
	(0.005)
Female	-0.170
	(0.148)
College Educated	-0.930***
	(0.169)
Household Income	-0.025
	(0.026)
Ideology (L-C)	0.931***
	(0.098)
Partisanship (D-R)	0.706***
	(0.041)
Born Again Christian	0.311*
	(0.181)
Intercept	-5.602***
	(0.448)
Num.Obs.	1821

* $p < 0.1$, ** $p < 0.05$, *** $p < 0.01$

Interpreting These Results

- Estimated natural logged odds of a Trump vote when all those things are 0 is about -5.602, but that person doesn't exist.
- Largest (absolute) effects are college education (-0.93), ideology (0.931), and partisanship (0.706).
- We don't appear to discern any effects of income or gender.

A Question

What is the largest effect on the white Trump vote in PA?

- Few/none of these variables share a common scale, so coefficient comparisons won't help.
- You can discern precision and discernibility from zero.
- You *cannot* say one is necessarily bigger than the other.

Why so?

- College education is binary, which (all else equal) drives up coefficient (and standard error)
- Age (for example) has 71 different values, which drives down coefficient (and standard error)

Use your head: we're talking about a partisan vote here (for president).

- Partisanship should be way more important than education, but it has more categories than college education.

Scaling Everything That's Not Binary

```
Penn %>%
    mutate_at(vars("age", "famincr","pid7na","ideo"),
              # r2sd() is in stevemisc
              list(z = ~r2sd(.))) %>%
    rename_at(vars(contains("_z")),
              ~paste("z", gsub("_z", "", .), sep = "_") ) -> Penn

M3 <- glm(votetrump ~ z_age + female + collegeed + z_famincr +
            z_ideo + z_pid7na + bornagain, data=Penn,
          family=binomial(link="logit"))

tidyM3 <- broom::tidy(M3)
```

Table 3: Predicting the White Trump Vote in 2016 (CCES, 2016)

	Did White PA Respondent Vote for Trump? (Standardized)
Age	0.323**
	(0.160)
Female	-0.170
	(0.148)
College Educated	-0.930***
	(0.169)
Household Income	-0.149
	(0.157)
Ideology (L-C)	1.987***
	(0.209)
Partisanship (D-R)	3.087***
	(0.179)
Born Again Christian	0.311*
	(0.181)
Intercept	0.392***
	(0.132)
Num.Obs.	1821

* p < 0.1, ** p < 0.05, *** p < 0.01

Interpretation

Notice what *didn't* change.

- Scaling the other variables doesn't change the binary IVs.
- Notice the *z*-value doesn't change either even as coefficient and standard errors change.

However, this regression table is much more readable.

- y-intercept is much more meaningful. It's natural logged odds of voting for Trump a non-born again, non-college educated white man of average/values/income.
- It suggests (which, use your head) that partisanship and ideology have the largest effects.

Conclusion

We're building toward an important point: *regression is akin to storytelling.*

- Tell your story well and get the most usable information out of what you're doing.

Some preliminary parlor tricks via Gelman:

- "Divide by 4": takes unintuitive logistic regression coefficients and returns upper bound predictive difference.
- Scaling by two SDs: provides preliminary comparison of coefficients (including binary inputs) and makes *y*-intercepts meaningful.

Table of Contents