Assignment 3: You'll Just Run 300,000 Regressions

PSCI 8357, Spring 2016
January 28, 2016

This assignment must be turned in by the start of class on **Thursday, February 4**. You must follow the instructions for submitting an assignment.

Main Task

In class this week, we discussed how replication with new data can help ameliorate the problems caused by publication bias and the statistical significance filter. But what can we do if new data isn't forthcoming?

The literature on economic growth faced exactly this problem in the early 1990s. Dozens of different variables had been found to be statistically significant correlates of growth, depending on the exact specification one used. But we can't wait another 50 years to replicate these studies with another half-century of growth data. To figure out which correlations were robust—i.e., which ones were not sensitive to a particular choice of specification—Levine and Renelt (1992) regressed growth on every conceivable combination of covariates (within limits). Sala-I-Martin (1997), in the brilliantly titled article "I Just Ran Two Million Regressions," refined their approach.

In this assignment, you will only run a little more than 300,000 regressions.[1] The application is a political science topic as prominent and bedeviling as that of growth rates in economics: the correlates of civil war onset. Your analysis will mirror that of Hegre and Sambanis (2006), who bring the millions-of-regressions approach to the civil war literature. You will be working with the data file `hs-imputed.csv`, a cleaned version of the Hegre and Sambanis (2006) data with missing values imputed.[2]

[1] Specifically, you will run $3 \times \binom{87}{3} = 317,985$ regressions.

[2] See the file `clean-and-impute.r` for how I cleaned up their replication data and imputed missing values. Because of the sheer number of variables involved relative to the time available to prepare a homework assignment, my approach was cursory and sloppy. In your own

The data is in country-year format, with the following variables:

- `country` and `year`: Self-explanatory.
- `warstns`: Indicator for whether a civil war began. This will be the response variable in all of your regressions.
- Three "core" covariates to include in every regression:
  - `ln_popns`: The natural logarithm of the country's population.
  - `ln_gdpen`: The natural logarithm of the country's GDP per capita.
  - `pt8`: $2^{t/8}$, where t is the number of years the country has been at peace.
- Eighty-eight "concept" variables whose association with civil war onset you will be testing. Descriptions of each appear in Table 1 of Hegre and Sambanis (2006).

You will select $J = 3$ of the 88 "concept" variables to test the robustness of their association with civil war onset. One of these must be `ehet`, the ethnic heterogeneity index; the other two are up to you. For each variable, we want to estimate the average probability of $\hat{\beta}_j > 0$, where the average is taken over the sampling distributions of each different regression specification. For each of these variables, $j = 1, 2, 3$, you will do the following:

- Consider the set of $M = \binom{87}{3}$ combinations of three covariates from the 87 other concept variables.
- For each $m = 1, \ldots, M$, you will:
  - Regress `warstns` on your chosen variable X_j, the three core covariates, and the m'th combination of other concept variables. (So every regression will have seven covariates, plus an intercept.)
  - Save the following quantities:
    - $\hat{\beta}_{jm}$: the estimated coefficient on X_j
    - $\hat{\sigma}^2_{jm}$: the estimated variance (squared standard error) of the coefficient on X_j
    - R^2_{jm}: the R^2 of the regression
- Using the results across each model, calculate the following:
  - A weight for each model, proportional to its R^2:

$$\omega_{jm} = \frac{R^2_{jm}}{\sum_{l=1}^{M} R^2_{jl}}$$

manuscripts, including the final paper for this class, you should be considerably more inquisitive and careful about missing values than I am in `clean-and-impute.r`.

– The average estimated coefficient,

$$\bar{\beta}_j = \sum_{m=1}^{M} \omega_{jm} \hat{\beta}_{jm}.$$

– The average estimated variance of the coefficient,

$$\bar{\sigma}_j^2 = \sum_{m=1}^{M} \omega_m \hat{\sigma}_{jm}^2.$$

– The so-called average p-value,

$$\bar{p}_j = \sum_{m=1}^{M} \omega_{jm} \Phi(0 \mid \hat{\beta}_j, \hat{\sigma}_j^2),$$

where $\Phi(0 \mid \mu, \sigma^2)$ is the probability of drawing a value less than zero from a normal distribution with mean μ and variance σ^2. This is *not* the average of the individual p-values, since direction matters.

In the last step, you're essentially answering the following: Suppose you drew a model at random, where the probability of choosing each model is proportional to its R^2. Then, suppose you drew a value of β_j from the estimated sampling distribution of the coefficients of this model. What is the probability that the value you drew is less than zero? A result close to 0 or 1 indicates a robust relationship. A result close to 0.5 indicates that the sign of the estimated coefficient is highly dependent on the particular specification—and thus the relationship is not robust.

Interpret your results. Which of the covariates you chose is robustly associated with civil war onset? Which are not? What are the limitations of this approach—what might we be missing by analyzing robustness this way?

Your results will not be exactly the same as in Hegre and Sambanis (2006). First, you are assuming a linear model and using OLS, whereas they assume a logistic model and use its maximum likelihood estimator. Second, missing values in the data have been imputed, so you do not need to worry about combinations of covariates that make the sample size too small.

Weekly Visualization Challenge

Compare the estimated distribution of draws of $\hat{\beta}_j$ across the three variables you chose to study.

Hints

Here are some R tips I found useful in completing this assignment myself. You don't have to use them, but I bet you will find them useful too.

Caching Output

It should take about 12 seconds to run a thousand regressions, which means about an hour to run 300,000.

```
300000 * (12 / 1000) / 60
```

```
## [1] 60
```

You don't want to have to spend an hour recompiling your R Markdown document every time you make a tiny change. I strongly recommend using the `cache = TRUE` chunk option for your major computations, as described in http://yihui.name/knitr/demo/cache/.

Creating Combinations

You can use the `combn()` function in R to generate all possible combinations of a particular length from a particular set. For example, here we have every combination of two letters from the first five letters in the alphabet.

```
combn(x = letters[1:5], m = 2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,] "a"  "a"  "a"  "a"  "b"  "b"  "b"  "c"  "c"  "d"
## [2,] "b"  "c"  "d"  "e"  "c"  "d"  "e"  "d"  "e"  "e"
```

Set Difference

Suppose you have two vectors `a` and `b`, and you want to retrieve every element of `a` that is not an element of `b`. You could do this the long way.

```
a <- c("a", "b", "c", "d", "e")
b <- c("b", "d")

a[!(a %in% b)]
```

```
## [1] "a" "c" "e"
```

Or you could just use the set difference function, `setdiff()`. Surely you remember set differences from the math boot camp.

```
setdiff(a, b)
```

```
## [1] "a" "c" "e"
```

This is especially useful if `a` and `b` have long and unwieldy names. While you're thinking about set operations, `union()` and `intersect()` work too.

```
union(a, b)
```

```
## [1] "a" "b" "c" "d" "e"
```

```
intersect(a, b)
```

```
## [1] "b" "d"
```

Selecting Variables with a Character Vector

You can use **dplyr**'s `select(one_of(var_names))` to select variables according to the character vector `var_names`.

```
library("dplyr")

fake_data <- data.frame(matrix(rnorm(30), ncol = 5))
names(fake_data) <- letters[1:5]
fake_data
```

```
##        a       b      c        d      e
## 1 -0.203 -1.3463  0.142 -0.00558 -1.047
## 2 -0.288 -0.0795  0.474 -0.53452 -1.266
## 3 -1.742  0.4160  1.149 -0.53433 -0.683
## 4 -0.132  0.5045  0.979 -1.34035  0.555
## 5  0.496  2.3698 -0.990  0.98617 -1.765
## 6 -1.086  0.4925 -0.333  0.75479 -0.372
```

```
var_names <- c("b", "d")
fake_data %>% select(one_of(var_names))
```

```
##         b        d
## 1 -1.3463 -0.00558
## 2 -0.0795 -0.53452
## 3  0.4160 -0.53433
## 4  0.5045 -1.34035
## 5  2.3698  0.98617
## 6  0.4925  0.75479
```

Regress on All Variables

Suppose you have a data frame with a bunch of variables in it, and you want to regress one of those variables on all the others. You can do this without writing out a long formula. Just use a formula like `y ~ .`, where `y` is the name of your response variable.

```
linear_fit <- lm(a ~ ., data = fake_data)
summary(linear_fit)
```

```
##
## Call:
## lm(formula = a ~ ., data = fake_data)
##
## Residuals:
##       1       2       3       4       5       6
##  0.1474 -0.2117  0.0623  0.0475  0.0821 -0.1276
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.3495     0.2541   -1.38     0.40
```

```
## b            -0.0323     0.1326   -0.24     0.85
## c            -2.3078     0.4484   -5.15     0.12
## d            -1.9323     0.4087   -4.73     0.13
## e            -0.2612     0.2238   -1.17     0.45
##
## Residual standard error: 0.309 on 1 degrees of freedom
## Multiple R-squared:  0.97,   Adjusted R-squared:  0.848
## F-statistic: 7.97 on 4 and 1 DF,  p-value: 0.259
```

Extract Regression Details

R makes it unnecessarily hard to extract the estimated standard errors and other interesting quantities from regression results. The **broom** package does the hard work for you. Just use the `tidy()` function to get a data frame with the information you want.

```
library("broom")
tidy(linear_fit)
```

```
##          term estimate std.error statistic p.value
## 1 (Intercept)  -0.3495     0.254    -1.376   0.400
## 2           b  -0.0323     0.133    -0.244   0.848
## 3           c  -2.3078     0.448    -5.147   0.122
## 4           d  -1.9323     0.409    -4.728   0.133
## 5           e  -0.2612     0.224    -1.167   0.451
```

The output is a data frame, which means we can use all our favorite **dplyr** tools on it.

```
tidy(linear_fit) %>%
    filter(term == "d") %>%
    select(estimate, std.error)
```

```
##   estimate std.error
## 1    -1.93     0.409
```

You can extract additional information, including the R^2, with the `glance()` function. Its output looks like a vector, but it's actually a one-row data frame.

```
glance(linear_fit)
```

```
##   r.squared adj.r.squared sigma statistic p.value df logLik AIC  BIC
## 1      0.97         0.848 0.309      7.97   0.259  5    3.9 4.2 2.95
##   deviance df.residual
## 1   0.0957           1
```

```
glance(linear_fit) %>% select(r.squared)
```

```
##   r.squared
## 1      0.97
```

References

Hegre, Håvard, and Nicholas Sambanis. 2006. "Sensitivity Analysis of Empirical Results on Civil War Onset." *The Journal of Conflict Resolution* 50(4): 508–35. http://www.jstor.org/stable/27638504.

Levine, Ross, and David Renelt. 1992. "A Sensitivity Analysis of Cross-Country Growth Regressions." *The American Economic Review* 82(4): 942–63. http://www.jstor.org/stable/2117352.

Sala-I-Martin, Xavier X. 1997. "I Just Ran Two Million Regressions." *The American Economic Review* 87(2): 178–83. http://www.jstor.org/stable/2950909.