Problem Set 2: Most Common Problems

POLS 8810

Spring 2025

Problem 1: Defining Your Model

It is vital to tell the reader what model you are estimating. This can be done solely in words (probably the most common approach) or with an equation that is then explained. You must do one of these. It is not ok to jump from a description of variables to some results. That leaves the reader to wonder where did these results come from? That question needs to be preempted.

If you take the "in words" approach, it is important to include all relevant information. For something simple like this, that is the estimator (e.g. I used OLS estimation to...) and the dependent and independent variables. In the future, in this class and beyond, this may include things like a justification for the estimator used, adjustments to the standard errors (and why those were made), and a variety of other modeling issues.

If you take the equation approach, it is not sufficient to provide the standard OLS equation ($y_i = \beta_0 + \beta_1 x_i + u_i$) and what each term means in a general sense. This is **NOT** what you are modelling. If you are going to take this approach, you need to give me **your** equation that is being estimated (e.g. $Republican\ Support_i = \beta_0 + \beta_1 Income_i + u_i$). Then tell me what is terms means in the context of **your** model. Note that in doing this, it is not necessary to talk about β_0 and u_i beyond saying that the former is an intercept term and the latter represents the residuals (see the minor points section of this handout). As with the "in words" approach, in the future, you will also need to provide additional information beyond the equation and what the terms mean.

Problem 2: Be Careful with How You Use Graphs

This problem has two parts, one minor (that nearly everyone did) and one much more significant that I highlighted in marginal notes on some papers.

2A: Including Unnecessary Figures/Tables

First, the minor issue. If a graph (or table) is not important enough to talk about, then do not include it. A lot of you had descriptive graphs of variables, but as the reader, I had no clue why they were there. What did they tell me that I needed to know? What was their contribution to the overall paper? The instructions to the assignment provide a good rule for research generally: "*While you should always [thoroughly examine your variables] as a standard step in the process any time you are working with data, is not necessary to include any of this in your write up (unless you feel that something you found at this stage is relevant to the primary analysis/analyses).*"

If a descriptive graph or table or a discussion of the central tendency or distribution of a variable is important in some way, please include it. However, in that discussion, it should be clear **why** it is important enough that you included it.

2B: Not Understanding Your Figures/Tables

Turning to the more major issue, when you include a graph, you need to know what it is. Several of you included a scatterplot of income and support for the Republican part with a fitted line and described it as a plot of your model estimates. However, that was **NOT** what you presented. This plot was not made with your model estimates, it was made with the actual data. In other words, this was a descriptive graph not a graph of model estimates. This is a very important difference and these two things have vastly different purposes.

Post-estimation graphs are great, intuitive ways to show substantive effects from a model's estimates. However, these are made from the stored results from the estimation of the model. What most of you did was to simply plot a scatterplot of your raw data (x_i and y_i) rather than your estimates (x_i and $\hat{y}_i$). Moreover, while the fitted line you used on this scatterplot is likely very similar to the line produced from your model estimates, *that is only true because this is a bivariate analysis.* This approach would **NOT** yield a similar estimated line of best fitted in a context with multiple independent variables as the model estimates for the relationship between a given independent variable and the dependent variable would be dependent on the other independent variables (as they are part of the full model) whereas fitting a line to a scatterplot of the raw data would git that line based solely on the bivariate relationship. **Please understand what a graph is before using it.**

Problem 3: Stop Truncating Variation!!!

I have no idea why so many of you thought it was a good idea to take a continuous variable, or even a ordinal one that approximated being continuous, and intentionally truncate the variation in the variable(s). All else being equal, *more* variation in our variables is good. Absent a very compelling reason (theoretical or related to the data), do not ever truncate variation.

Problem 4: What Goes in Replication Code?

I am not going to critique the aesthetics of your code. I do not write pretty code, so I do not expect yours to be perfect examples of good coding practices. However, there are a few issues that go beyond aesthetics and impact the functionality of your replication code. These are things you need to work on.

The biggest issue is noise in your code. many of you provide a ton of code that does a lot of things that are **NOT** in your write up. This is not only unnecessary, but it is antithetical to the purpose of replication code. Good researchers do a lot of things to help us understand our data prior to analysis and a lot of additional things for robustness checks post-analysis, but unless these things go in the paper, they do not go in the replication code. In other words, every line of code you write is not replication code.

So what does go in a replication code file? The answer is in the name. You include precisely all of the things necessary to replication what is in the paper, nothing more and nothing less. Here are some general guidelines of what to include:

- Comment your code! Best practices are to comment basically everything, but I am not going to ask your to do that (although you probably should). I only ask that you comment the specific things that appear in your paper, telling me where they appear in the paper.
- All packages needed to do everything in the code.
- Commands to read in data from a publicly available repository or data provided.
- Any variable transformations that alter the data you read into R (e.g. recoding variables, renaming variables, etc).
- Any data summary commands that produce things that are included in the paper. This would include both simple commands like `summary()` when used to get summary stats discussed in the text or table and more complex things like descriptive graphs that are included in the paper.

- Any analyses that are reported in the paper (even if they are not a primary analysis and only mentioned briefly, e.g., as a "robustness check").
- Any graphs/figures that are reproduced in the paper.
- Any other preliminary or post-estimation tests, analyses, etc that are included or discussed in the paper. This could include anything from tests of model assumptions to post-estimation robustness checks.

Here are some things to **never** include in replication code:

- Any packages **not** needed to do something in the code.
- Any commands used to view data.
- Any data summary commands that do not produce things used in the paper. This would include both simple commands like `summary()` and more complex things like descriptive graphs that are **not** included in the paper.
- Any commands that **only** work with RStudio. (Remember you are making replication code that should be useable by any R user.)
- Any analyses that are **not** reported anywhere in the paper.
- Any graphs/figures that are **not** reproduced in the paper.
- Any other preliminary or post-estimation tests, analyses, etc that are **not** included or discussed in the paper.

Another best practice in writing replication code is to never include any code in replication files that saves files to a local machine (e.g. graphes, TEXcode for tables, etc). Obviously you want to be able to save your files as needed, but this are not something you want in the replication code. On the other hand, some of you have code in your replication file that makes a graph but does not create an object. This is also not good. How do you even access the graph you have created later in your R session? The replication code should always produce the object, not but save it locally. You should remove any code that saves files to a local directory (or, at least, comment them out) before sharing replication code.

Similarly, you should not ever include a full or partial path directory in your replication code. My person approach is to set my local working directory prior to beginning a session in R to whatever folder everything in a given project will be located. If I have to reference a subfolder with a partial path directory (e.g. if my data is in a subfolder labeled data), I would just cut that partial path directory out of the replication code I share with others but

leave it in my personal code file (e.g. I might have `data <-read_dta("data/anes2000TS.dta")` in my personal replication file, but would only put `data <- read_dta("anes2000TS.dta")` in a file I was going to share).

Finally, there are a lot of stylistic issues with your code, including some pretty basic stuff related to naming conventions. While these things not essential to be able to meet the course requirements, or even to do good work, they are bad habits that it is better to fix before they become permanent fixtures of your coding practices. Ozlem has provided some resources to help you with this on the course GitHub page she manages.

Very Common, But More Minor, Problems

- While much improved from Problem Set 1, many of you still need to go back and reread Problem 1 in the handout I gave summarizing the major problems with that problem set. Here are a couple of rules of thumb to help. First, if you do not start with a clear variable name and a description of how *your* variable is coded, you are doing it wrong. Second, you should pretty much never mention R. If you are talking about software, you are probably doing it wrong (there are exceptions, but they are largely for things beyond the scope of this class). Third, you should pretty much never discuss commands you would use in R or computer code more generally. Instead, you should talk about what you are doing conceptually, not what you are asking the computer to do in order to accomplish the conceptual goal (again, there are exceptions, but they are largely for things beyond the scope of this class).

- If a number is in the table, it almost never needs to be repeated in the text. You do not need to say "the coefficient estimate was X," "the standard error was X," etc. While there are exceptions, this is generally to be avoided. (Note: this doe NOT mean that you shouldn't use the coefficient value in substantively interpreting its meaning (e.g. "my model predicts that for every one unit change in X we should expect a β_1 unit change in Y"), it just means it is not necessary to say something like "the coefficient estimate for X was β_1" since that is already clearly in the table.

- Do **NOT** interpret the intercept. There are some very specific exceptions to this, but in general, the intercept is substantively meaningless.

- **Do NOT report multiple levels of statistical significance in a table!!!** Doing so shows a basic failure to understand what statistical significance means conceptually in the context of null hypothesis significance testing!

- **Do NOT interpret the R^2 as if it was substantively meaningful!**

- Please stop using "utilize" when you mean "use."

- Footnote markers go outside punctuation or closing quotation mark. For example, words[1]. and "words[1]." are incorrect, whereas words.[1] and "words."[1] are correct.
- A couple of you used μ_i to represent the error term. Don't do this. μ_i is pretty much exclusively used in statistics for a mean. The two most common ways to denote the error term are u_i (what I have, and will continue to, use in lecture slides) and ϵ_i.