Power Analysis Through Simulations in STATA: A Step-by-Step Guide

Campos-Mercade, Pol

2024

Document Version:
Other version

Working Paper 2024:4

Department of Economics
School of Economics and Management

Power Analysis Through Simulations in STATA: A Step-by-Step Guide

Pol Campos-Mercade

August 2024

LUND UNIVERSITY

Power analysis through simulations in Stata: a step-by-step guide

Pol Campos-Mercade*

August 2024

You want to run an experiment where you anticipate finding a treatment effect. How large should your sample size be to have a reasonable chance of detecting significant results? In this manuscript, I present and explain the Stata code I use to address this question. The code uses simulations to conduct power analyses, offering a flexible alternative to the commonly used analytical tools. Unlike traditional methods, this approach can accommodate any experimental design and statistical test that Stata supports. The code is straightforward, user-friendly, and can be used effectively with minimal coding experience.

Keywords: Power analysis, Simulations, Stata.
JEL classification: C15, C90.

Update: *This manuscript was originally drafted in 2018, and I never anticipated it would reach such a wide audience or even become a part of experimental courses. I have now made it available as a Working Paper. If you find the information here useful for your work, I would appreciate it if you could cite this paper. Doing so would help to share this resource with others and potentially benefit more experimentalists.*

* Lund University, Department of Economics, Tycho Brahes väg 1, Lund, Sweden. E-mail: pol.campos@nek.lu.se.

Power analysis through simulations in Stata: a step-by-step guide

Pol Campos-Mercade*

August 2024

You want to run an experiment where you anticipate finding a treatment effect. How large should your sample size be to have a reasonable chance of detecting significant results? In this manuscript, I present and explain the Stata code I use to address this question. The code uses simulations to conduct power analyses, offering a flexible alternative to the commonly used analytical tools. Unlike traditional methods, this approach can accommodate any experimental design and statistical test that Stata supports. The code is straightforward, user-friendly, and can be used effectively with minimal coding experience.

Keywords: Power analysis, Simulations, Stata.
JEL classification: C15, C90.

Update: *This manuscript was originally drafted in 2018, and I never anticipated it would reach such a wide audience or even become a part of experimental courses. I have now made it available as a Working Paper. If you find the information here useful for your work, I would appreciate it if you could cite this paper. Doing so would help to share this resource with others and potentially benefit more experimentalists.*

* Lund University, Department of Economics, Tycho Brahes väg 1, Lund, Sweden. E-mail: pol.campos@nek.lu.se.

***Disclaimer**: Using simulations to do power analyses has been used for a long time (Feiveson 2002; Arnold et al., 2011). There are even Stata packages (powersim, Luedicke 2013; powerBBK, Bellemare et al. 2016) and R packages (SIMR, Green and MacLeod 2016) to do specific types of power analyses through simulations. Despite the widespread use of these methods among statisticians, my experience is that most experimental economists still rely on analytical tools for power analyses. While these tools are effective for simple experimental designs, they lack the flexibility to accommodate a wide range of experiments and statistical tests. The purpose of this manuscript is to share the Stata code I frequently use for power analyses, which many of my colleagues have found to be particularly useful.*

1. Introduction

Conducting a power analysis is a fundamental aspect of any experimental design. For most basic designs, equations are available to perform power calculations analytically (see, for example, List et al. 2011). [1] However, many more complex designs lack straightforward analytical solutions or are difficult to derive. These complexities may include factors such as clustering, treatment interactions, covariates, or non-normal distributions requiring non-parametric tests. In such cases, simulations offer a powerful and flexible tool for conducting your power calculations.

In this manuscript, I share the code I use to perform power calculations. The code is written in straightforward programming language, with the aim that any reader with basic programming knowledge can easily understand and apply it. It is also highly flexible, capable of accommodating any experimental design or statistical test that the reader wishes to run.

[1] G-Power (Faul et al. 2007) is a very useful tool to perform such basic power calculations. You can download it here: http://www.gpower.hhu.de.

In Section 2, I define power and explain why conducting power analyses is important for your research. Section 3 provides a summary of how the code works. Section 4 offers a simple example of using the code, for which analytical solutions exist. In Section 5, I discuss a more complex example, where analytical solutions are not available.

2. Power

This section offers an overview of the concept of power. If you're already familiar with this topic, feel free to skip ahead.

2.1 What is power?

Power is the probability of detecting an effect when the effect actually exists. In other words, if the treatment we are testing has a real effect, power indicates the likelihood that our experiment will yield statistically significant results for that effect.

Experimenters typically use power calculations for two main purposes. The first is to determine the sample size required to have a reasonable chance of rejecting the null hypothesis (no effect) if the alternative hypothesis (an actual effect) is true. To calculate this sample size, you need to specify an effect size for your alternative hypothesis—essentially, how large you expect the treatment effect to be. There are three primary ways to choose this effect size: based on your best estimate, from

results in previous research[2], or by determining the minimum effect size that would be meaningful to detect. [3]

Second, researchers may conduct a power analysis to determine the minimum effect size that can be reliably detected given a fixed sample size. This is particularly useful in situations where the sample size is predetermined, such as the number of students in a class. By calculating how large the true effect must be to achieve, for instance, an 80% chance of finding a significant result, researchers can assess whether their experiment is well-designed or overly ambitious. This type of analysis is important to ensure that the experiment's objectives are realistic and achievable.[4]

2.2 Why is it important to conduct power analyses?

Conducting power analyses is important for several reasons:

1. It increases the chances of obtaining statistically significant results. With low power, such as 40%, there's a 60% chance of failing to detect a true effect, leading to a null result. Recognizing this early and increasing your sample size to achieve, for instance, 80% power, will improve your ability to detect true effects and boost the credibility of your research.

[2] Some researchers also use results from pilot experiments to specify the expected effect size. However, I am often skeptical towards this approach. Pilot experiments are typically conducted with small sample sizes, resulting in noisy estimates that can be less reliable than informed guesses. Relying too heavily on these estimates can lead to significant errors. That said, pilot experiments are very valuable for understanding the data-generating process, which is also necessary for conducting accurate power analyses.

[3] Most treatments that we try will have *some* true effect. However, if this effect is 0.001 standard deviations, in most cases this will not be of any practical relevance.

[4] During the first year of my PhD studies, when I had little understanding of power analysis, I conducted a class experiment with 200 students. I divided them into a control group and two treatment groups of about 65 students each. I didn't find any significant results. Later, when I calculated the effect size that my study was actually powered to detect, I realized that it was far too large. Had I performed a power analysis beforehand, I would have likely opted for just one treatment group, giving me a better chance of detecting something meaningful. As it stands, because my study lacked the power to detect a reasonable effect size, even the null results aren't particularly useful. The data now sits unused in my drawer since, obviously, no one cares about an underpowered null.

2. It optimizes resource use in experiments. Experiments are often costly and time-consuming. A power analysis ensures that your sample size is appropriate for detecting the expected effect, preventing you from unnecessarily overspending on a too large sample.
3. It strengthens the credibility of null results. A null result from an experiment with only 20% power is uninformative—it does not clarify whether the effect is truly absent or simply undetected. However, if you can demonstrate that your study had 80% power to detect a reasonable effect, your null result becomes much more meaningful.
4. It improves the precision of your estimates, making the treatment effect in your experiment more robust and meaningful.
5. It benefits science. Significant results from low-power studies are less likely to reflect true effects and are less likely to be replicated. This contributes to replication crises, undermining the credibility of scientific knowledge.

3. Method

The simulation requires the following steps:

1. Set the sample size for the simulated experiment.
2. Assign a baseline value to each observation, representing what is expected in the *absence* of a treatment effect. You can choose this baseline value using one of the following two options:
    a. Simulate a distribution using Stata's tools (e.g., the command `gen x = rnormal()`). To determine which distribution to draw from, you can base your choice on previous results, a dataset of the subjects you plan to experiment with, a pilot study, or just your best guess.
    b. Use data that already exists. If available, you can bootstrap this data to populate your simulated sample. For instance, if you want to test whether an intervention affects students' GPA, you can use data from a previous cohort. If you have data for 100 students but need a

simulated sample size of 200 observations, you can use bootstrap with replacement and randomly assign the data from each of the 100 students to each of the 200 observations.

3. Randomly assign each observation to each treatment based on the proportion of treated vs control subjects that you would like to simulate.
4. For those who have been assigned to the treatment group, add the expected treatment effect.
5. Run your test on the created dataset and store the p-value of the test.
6. Repeat steps 2-5 many times.
7. Count the number of instances where the p-value is less than 0.05. This represents your statistical power. For example, if you repeated steps 2-5 a thousand times and found that the p-value was below 0.05 in six hundred cases, you can conclude that you have 60% power to detect the effect introduced in step 4.

I will now present two examples showing how to apply this method. Please note that the code provided is designed for clarity and ease of understanding rather than efficiency or elegance. It is written to be accessible for those with basic coding experience.

4. Example 1

Now, let's apply this method to a straightforward example. Suppose you design a lab experiment where subjects are tasked with completing a task as quickly as possible. You want to test whether incentivizing them to finish faster—by offering higher payments for quicker completion—will reduce the time it takes them to complete the task. For this example, assume the following:

1. You know (or guess) that the time required for subjects to complete the task follows a normal distribution with a mean of 100 seconds and standard deviation of 20 seconds (note that in this case a lognormal distribution seems

more reasonable, but we will use a normal distribution for the sake of the example).

2. You anticipate that the true effect of incentivizing subjects will reduce their completion time by 5 seconds.
3. You plan to have an equal number of subjects in both the control and treatment groups.

How many subjects do you need in your study to achieve 80% power? In other words, how many subjects are required for your experiment to reject the null hypothesis with an 80% probability if the true effect is indeed a 5-second reduction? Let's dive into the coding to find out!

```
************************************
* This Stata code is for Stata 15. It should work also work in most other Stata
versions (sometimes requiring minor modifications)
************************************
* This code performs a power analysis for an experiment in which: the observations
from the control group follow a normal(100,20); the true treatment effect is -5;
the significance level is 0.05; the sample is 200, equally divided between control
and treatment group. We will calculate the power of this experiment to detect a
significant result.
************************************
clear
set matsize 1000
mat estimates = J(1000,1,.)
* Creates a matrix of 1000 rows by 1 column. In each row, we will store the p-value
of each simulation.
local subjects=200
local teffect=5
* Defines the number of subjects in our experiment and the treatment effect.
quietly forvalues j=1(1)1000 {
* Repeats the following code 1000 times.
clear
set obs `subjects'
gen id=_n
* Sets 200 observations and assign an ID to each one of them from 1 to 200.
gen treatment=0
replace treatment=1 if id >= `subjects'/2
* Assigns half of the subjects to the treatment group.
```

```
gen time=rnormal(100,20)
replace time=time-`teffect' if treatment==1
* Assigns an observation to each of the subjects. Those who are treated perform the
task 10 seconds earlier.
ttest time, by(treatment)
scalar pvalue = r(p)
* Tests differences between the control and treated group. Stores the p-value of
the test.
matrix estimates[`j',1] = pvalue
* Adds the p-value of the test in the row number "j" of column 1 on the 1000x1
matrix that we created.
noisily display `j'
* Shows you on which simulation you are at (personal preference)
}
* This experiment is repeated 1000 times.
svmat estimates, names(pvalues)
* Retrieves (and adds to our dataset) the 1000x1 matrix with the p-values of all
the simulated experiments.
gen significant=0
replace significant=1 if pvalues<0.05
* Creates a variable with value 1 if the experiment was significant.
ci means significant
* It displays the percentage of experiments in which the test was significant (power)
and its confidence interval. In my case, I had 41% power.
* Now you can play around with the number of observations and true treatment effect
to see how power changes with a different sample size and treatment effect.
```

5. Example 2

Let us consider a more complex scenario. Suppose you want to test whether incentivizing university students to achieve a specific grade improves their GPA. You plan to conduct this test during the students' second semester, allowing you to control for their first-semester GPA. For this example, assume the following:

1. There are 1,000 students in this cohort, and your task is to decide how many of them to incentivize. Therefore, your goal is not to determine the sample size, but rather to decide what percentage of students to treat.
2. You have data on the first-semester GPA for all 1,000 students in this cohort.
3. You also have data on the first- and second-semester GPA for 1,000 students from the previous cohort.
4. Students are divided into 4 groups of 250, with the possibility of GPA fixed effects in the second semester based on these groups.
5. The treatment effect is expected to vary among students, drawn from a normal distribution. Since a treatment effect of less than 2 (on average) would not be considered meaningful, we will conduct a power analysis assuming that the treatment effect follows a normal distribution with a mean of 2 and a standard deviation of 2.
6. We plan to perform three tests: a t-test, a Wilcoxon rank-sum test, and a class fixed-effects regression that controls for the students' first-semester GPA.

The question is: how many students should you incentivize to achieve 80% power? To determine this, we will use the dataset from the previous cohort and simulate 1,000 experiments. In each simulation, we will randomly select students to receive a hypothetical treatment effect added to their second-semester GPA. We will then calculate how often each test yields a significant result (i.e., the power of the experiment). Let's code this!

```
************************************
* This Stata code is for Stata 15. It should work also work in most other Stata versions (sometimes requiring minor modifications)
************************************
* This code performs a power analysis for an experiment in which we test whether incentivizing students boosts their GPA. We have: 1000 students to experiment with from which we know their first semester GPA; 1000 observations about the first and second semester GPA of students in the previous cohort (which go from 0, lowest grade, to 100, highest grade); the true treatment effect is distributed following a normal (2,2); there are four class groups with class fixed effects. Out of 1000 students, we have to decide how many students to incentivize to have 80% power.
************************************
clear
set matsize 1000
mat estimates = J(1000,3,.)
* Creates a matrix of 1000 rows by 3 columns. In each row, we will store the p-value of each simulation. In each column, we will store the p-value of each different test.
* Now we would use the dataset of the previous cohort, in which each row corresponds to each student and it shows the students' first semester GPA (gpa1), second semester GPA (gpa2) and class group. Instead of importing a dataset, in this example we generate it (and we imagine it is real)
set obs 1000
gen id=_n
gen classgroup=.
replace classgroup=1 if id<=250
replace classgroup=2 if id>250 & id<=500
replace classgroup=3 if id>500 & id<=750
replace classgroup=4 if id>750 & id<=1000
scalar fe1 = rnormal(0,10)
scalar fe2 = rnormal(0,10)
scalar fe3 = rnormal(0,10)
scalar fe4 = rnormal(0,10)
gen gpa1=rnormal(50,15)
gen gpa2r=rnormal(50,15)
gen gpa2=0.7*gpa1+0.3*gpa2r+fe1
replace gpa2=0.7*gpa1+0.3*gpa2r+fe2 if id>250 & id<=500
replace gpa2=0.7*gpa1+0.3*gpa2r+fe3 if id>500 & id<=750
replace gpa2=0.7*gpa1+0.3*gpa2r+fe4 if id>750 & id<=1000
drop gpa2r
* Generates the fake dataset in which gpa1 explains 70% of gpa2 and there are 4 class groups with their own fixed effects.
local treatedsubjects=500
local teffectscalar=1
```

```
* Defines the number of subjects that we are going to treat in this simulation and
the treatment effect (these are the values to play around with).
quietly forvalues j=1(1)1000 {
* Repeats the following code 1000 times.
capture drop randomnumber
capture drop treatment
capture drop teffect
capture drop idnew
* We drop variables from the previous loop, if there are any.
gen randomnumber=runiform()
sort randomnumber
gen idnew=_n
* We generate a random number for each observation to sort all observations randomly.
This is to randomly allocate the treatment (although, obviously, in this kind of
design it would be better to stratify).
gen treatment=0
replace treatment=1 if idnew <= `treatedsubjects'
* Assigns the treatment to the number of subjects that we previously set as treated.
gen teffect=rnormal(`teffectscalar',2)
replace gpa2=gpa2+teffect if treatment==1
* Assumes that the treatment effect is normally distributed around what we decided
in "teffectscalar" with a standard deviation of 2 (contrary to the previous example,
we no longer assume that the treatment effect is constant across all subjects).
ttest gpa2, by(treatment)
scalar pvalue1 = r(p)
matrix estimates[`j',1] = pvalue1
* Performs a t-test and stores it in the first row of the matrix.
ranksum gpa2, by(treatment)
scalar pvalue2=2 * normprob(-abs(r(z)))
matrix estimates[`j',2] = pvalue2
* Performs a Wilcoxon rank sum test. The pvalue for this test has to be computed
with the formula above (cannot be extracted with r(p) as with the t-test). Then
stores it in the second row of the matrix.
areg gpa2 gpa1 treatment, absorb(classgroup)
       local q = _b[treatment]/_se[treatment]
       scalar pvalue3 = 2*ttail(e(df_r),abs(`q'))
matrix estimates[`j',3] = pvalue3
* Performs a regression with class fixed effects and controlling for gpa1. The
pvalue for "treatment" is extracted from the formula above. Then stores it in the
third row of the matrix.
replace gpa2=gpa2-teffect if treatment==1
* Recall that we added the treatment effect to the dataset to make our test. For
each simulation, we want to start from the basic dataset and the treatment effect
```

```
to be different. So now we take the treatment effect away so that in the next loop
we start with the same dataset as before.
noisily display `j'
* Shows you on which simulation you are at (personal preference)
}
* This experiment is repeated 1000 times.
svmat estimates, names(pvalues)
* Retrieves (and adds to our dataset) the 1000x3 matrix with the p-values of all
the simulated experiments.
gen significant1=0
replace significant1=1 if pvalues1<0.05
* Creates a variable with value 1 if the test was significant.
gen significant2=0
replace significant2=1 if pvalues2<0.05
gen significant3=0
replace significant3=1 if pvalues3<0.05
ci means significant1
ci means significant2
ci means significant3
* It displays the percentage of experiments in which the test was significant (power)
and its confidence interval for each of the tests. In my example, with 200 treated
students, I got 13% power with the t-test, 12% power with the rank sum test, and
80% power with the regression test. If we wanted the t-test or the rank sum test to
be our main test, we do not have enough power with 200 incentivized students. But
if we want the regression test to be our main test, then we do have 80% power to
find a significant result.
```

6. Conclusion

Power analyses are useful and should be conducted before any experiment. No matter the complexity of the experimental design, power analyses can be easily conducted through simulations. This paper provides a hands-on, hopefully helpful example of how you can easily apply this method.

References

Arnold, B. F., Hogan, D. R., Colford, J. M., & Hubbard, A. E. (2011). Simulation methods to estimate design power: an overview for applied research. BMC medical research methodology, 11(1), 94.

Bellemare, C., Bissonnette, L., & Kröger, S. (2016). Simulating power of economic experiments: the powerBBK package. Journal of the Economic Science Association, 2(2), 157-168.

Feiveson, A. H. (2002). Power by simulation. Stata J, 2(2), 107-124.

Green, P., & MacLeod, C. J. (2016). SIMR: an R package for power analysis of generalized linear mixed models by simulation. Methods in Ecology and Evolution, 7(4), 493-498.

List, J. A., Sadoff, S., & Wagner, M. (2011). So you want to run an experiment, now what? Some simple rules of thumb for optimal experimental design. Experimental Economics, 14(4), 439.

Luedicke, J. (2013). Powersim: simulation-based power analysis for linear and generalized linear models. In 2013 Stata Conference (No. 13). Stata Users Group.