**A/B Tests, Privacy, and Online Regression**

***How to run experiments without storing individual-level data***

**AB tests**, a.k.a. [randomized controlled trials](https://en.wikipedia.org/wiki/Randomized_controlled_trial), are widely recognized as the gold standard technique to compute the **causal** effect of a treatment (a drug, ad, product, …) on an outcome of interest (a disease, firm revenue, customer satisfaction, …). The procedure consists in randomly splitting a set of subjects (patients, users, customers, …) into a treatment and a control group and giving the treatment to the treatment group. The **randomness** ensures that the expected difference between the two groups is caused by the treatment.

One potential **privacy concern** in A/B tests is that one needs to store data about many users for the whole duration of the experiment in order to estimate the effect of the treatment. This is not a problem if we can run the experiment instantaneously, but it can become an issue when the experiment duration is long. In this post, we are going to explore one solution to this problem: **online regression**. We will see how to estimate (conditional) average treatment effects and how to do inference, using both the central limit theorem and bootstrapping, one observation at a time storing only aggregate information.

⚠️ I have omitted the **algebra** behind some of the equations. If you want to see more background algebra, let me know and I will happily add another section.

**A Simple Example**

Suppose we were a **fin-tech company**. We have designed a new user interface (UI) for our mobile application and we would like to understand whether it slows down our transactions. In order to estimate the causal effect of the new UI on transaction speed, we plan to run an **A/B test** or randomized controlled trial: we split users into two groups at random, to one group we show the new UI, and we compare the average transaction speed across the two groups.

We have one major problem: we cannot store transaction-level information for **privacy reasons**. Moreover, we can’t run the experiment in one shot since we observe the transactions only when they are made. What can we do?

First, let’s have a look at the **data**. I import the data generating process dgp\_credit() from [src.dgp](https://github.com/matteocourthoud/Blog-Posts/blob/main/notebooks/src/dgp.py) and some plotting functions and libraries from [src.utils](https://github.com/matteocourthoud/Blog-Posts/blob/main/notebooks/src/utils.py). To include not only code but also data and tables, I use [Deepnote](https://deepnote.com), a Jupyter-like web-based collaborative notebook environment.

First, I generate the whole dataset. We will then investigate how to perform the experimental analysis in case the data were arriving dynamically.

We have information on 100 users, for whom we observe whether they were randomly assigned the newUI, their connection speed and their transfer speed.

First, let's estimate the treatment effect by regressing the outcome of interest (transfer speed) on the treatment indicator (newUI). Randomization ensures that the coefficient of newUI is an **unbiased** estimate of the causal treatment effect.

The coefficient of newUI is positive (6.5008) but not statistically significant (p=0.113).

We suspect that connection speed also impacts the transfer speed and conditioning the analysis on it might increase its power. Let’s run the same regression adding log(connection) as a **covariate**.

Indeed, the estimated coefficient has not changed much, but the standard errors have decreased and the coefficient is now **statistically significant** at the 5% level (p=0.031).

In order to understand how we can run linear regression one data point at a time, we first need a brief linear algebra recap.

First of all, let’s define *y* the dependent variable, transfer speed in our case, and *X* the explanatory variable, the newUI indicator, the log(connection) speed, and a constant term.

The OLS estimator is given by

https://miro.medium.com/max/700/1*Vvrc31rozJw54QlgPMkwnA.png

OLS estimator formula, image by Author

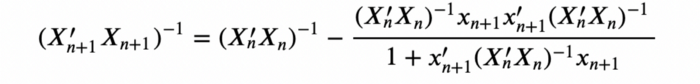
Indeed, we get the exact same number as with the smf.ols command!

Can we compute the OLS coefficient *β* **one observation at a time**?

The answer is yes! Assume we had *n* observations and we just received the *n+1*th observation: the pair (xₙ₊₁, yₙ₊₁). In order to compute *β̂ₙ₊₁,* we need to have stored only two objects in memory:

* *β̂ₙ,* the previous estimate of *β*
* (Xₙ’Xₙ)⁻¹, the previous value of (X’X)⁻¹

First of all, how do we update (X’X)⁻¹?



Updating rule for (X’X)⁻¹, image by Author

After having updated (X’X)⁻¹, we can update *β̂*.

https://miro.medium.com/max/700/1*yahgTOHmRPBVgYeOHXRQ6w.png

Updating rule for β*, image by Author*

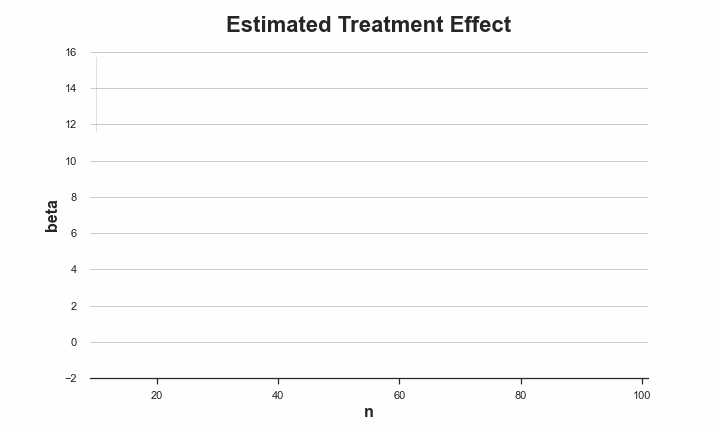
Note that this procedure is not only privacy-friendly but also **memory-friendly**. Our dataset is a 100×4 matrix while (X’X)⁻¹ is a 3×3 matrix and *β* is a 3×1 matrix. We are storing only 12 numbers instead of up to 400!

We are now ready to **estimate** our OLS coefficient, one data point at a time. However, we cannot really start from the first observation, because we would be unable to invert the matrix X’X. We need at least *k+1* observations, where *k* is the number of variables in *X*.

Let’s use a **warm start** of 10 observations to be safe.

We got exactly the same coefficient! Nice!

How did we get there? We can plot the evolution of our estimate *β̂* as we accumulate data. The dynamic plotting function is a bit more cumbersome, but you can find it in [src.figures](https://github.com/matteocourthoud/Blog-Posts/blob/main/notebooks/src/figures.py).



As we can see, as the number of data points increases, the estimate seems to become less and less volatile.

But is it really the case? As usual, we are not just interested in the point estimate of the effect of the newUI on spending, we would also like to understand how precise this estimate is.

**Inference**

We have seen how to estimate the treatment effect “online”: one observation at a time. Can we also **compute the variance** of the estimator in the same way?

First of all, let’s review what the variance of the OLS estimator looks like. Under baseline assumptions, the variance of the OLS estimator is given by:

https://miro.medium.com/max/700/1*-qXhBCLCdljV2IqJsBDb_w.png

OLS estimator’s variance formula, image by Author

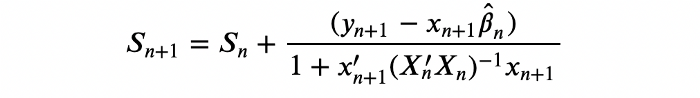
where *σ̂²* is the variance of the residuals *e=(y−X’β̂)*.

The regression table reports the standard errors of the coefficients, which are the squared roots of the diagonal elements of *Var(β̂)*.

Let’s check that we would indeed obtain the same numbers using matrix algebra.

Indeed, we get exactly the same numbers!

We already have a method to update online one part of the variance of *β̂:* (X’X)⁻¹*.* How do we update σ̂²? This is the formula to update the sum of squared residuals S.



Updating rule for S, image by Author

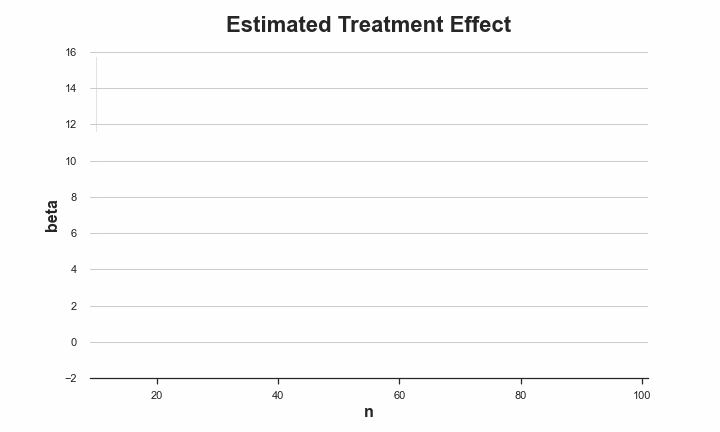
In order to get the residual variance *σ̂²* from the sum of squared residuals *S*, we need to divide by the degrees of freedom: n–k = 100–3.

Note that the order is important! *S* is computed using the old values of (X’X)⁻¹ and *β̂* so it has to be updated first.

We can now compute both *β̂* and its estimated variance one observation at a time.

We indeed got the same result!

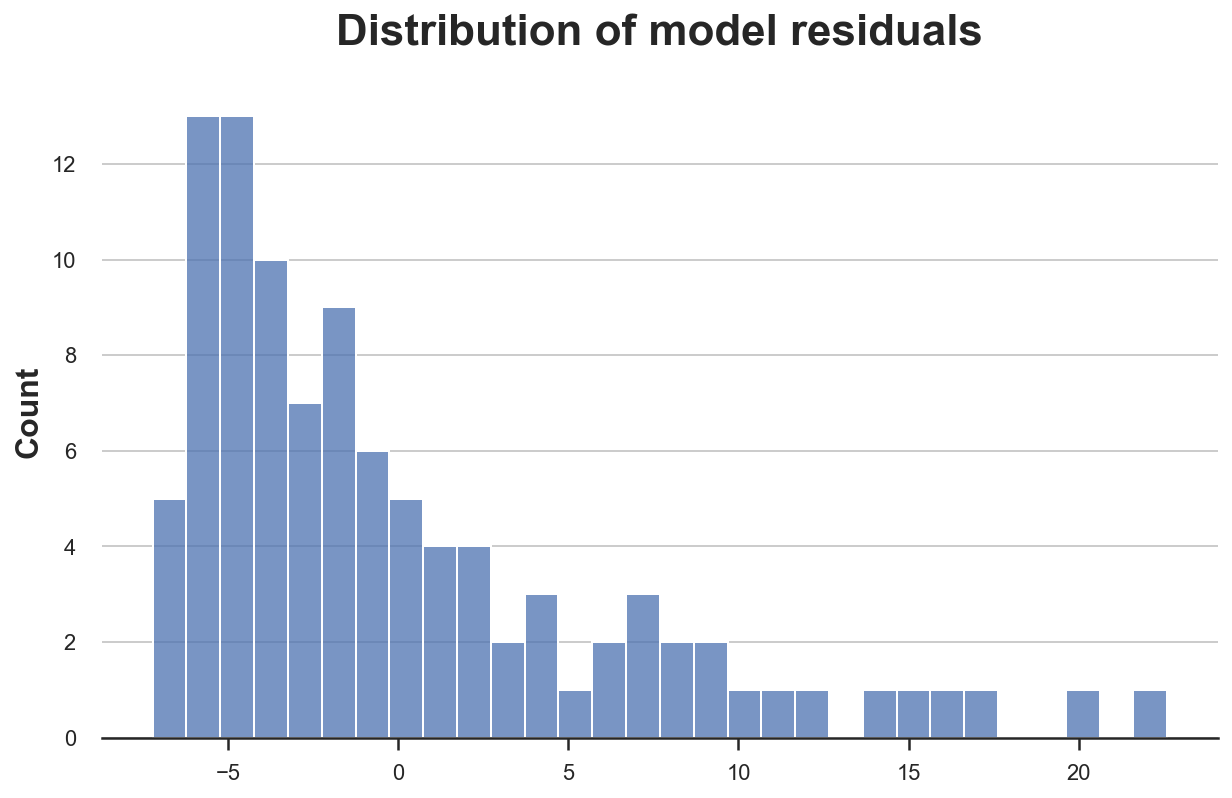
As before, we can **plot the evolution** of the estimate of the OLS coefficient over time, augmented with a confidence band of plus/minus one standard deviation.



As we can see, the estimated variance of the OLS estimator indeed decreases as the sample size increases.

**Bootstrap**

So far, we have used the asymptotic assumptions behind the Central Limit Theorem to compute the standard errors of the estimator. However, we have a particularly small sample. We further check the empirical distribution of the model **residuals**.

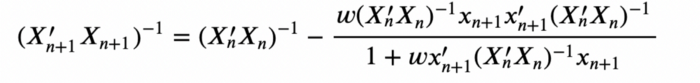


The residuals seem to be particularly **skewed**! This might be a problem in such a small sample.

One alternative to large sample theory is **the bootstrap**. Instead of relying on the [central limit theorem](https://en.wikipedia.org/wiki/Central_limit_theorem), we approximate the distribution of our estimator by re-sampling our dataset with replacement. Can we bootstrap online?

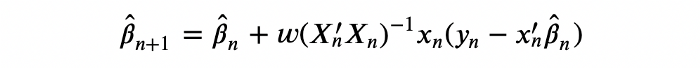
The answer is once again yes! The key is to **weight each observation** with an integer weight drawn from a Poisson distribution with mean (and variance) equal to 1. We repeat this process multiple times for each observation and we store the respective intermediate estimates. Instead of having a single intermediate estimate of *β* at any point in time, we will have *K*, the number of bootstrapped samples.

The updating rules for (X’X)⁻¹ and *β̂* become



Updating rule for (X’X)⁻¹ with bootstrap weights, image by Author

and



Updating rule for *β̂* with bootstrap weights, image by Author

where *w* are Poisson weights.

We can now run the online estimation. We bootstrap *K=1000* different estimates of *β̂*.

We can estimate the standard deviation of the treatment effect by computing the standard deviation of the vector of bootstrapped coefficients.

The estimated standard errors are slightly different from the previous values of [5.05, 3.01, 1.49], but not very far apart.

Lastly, some of you might have wondered “*why sampling discrete weights and not continuous ones?*”. Indeed, we can. This procedure is called the **Bayesian Bootstrap** and you can find a more detailed explanation [here](https://towardsdatascience.com/6ca4a1d45148).

**Conclusion**

In this post, we have seen how to run an experiment without storing individual-level data. How are we able to do it? In order to compute the average treatment effect, we do not need every single observation but it’s sufficient to store just a more \**compact representation* of it.

This procedure is not only privacy-friendly but also **memory efficient** since the inverted score matrix (X’X)⁻¹ and the estimated coefficient *β̂* usually have a much smaller dimension than the dataset (as long as the dimension of *X* is small, which is probably the case in randomized controlled trials).