Lab 5: August 7, 2019

Most of lab time today will be set aside for work on your final project. One thing you've probably noticed working on the project is how big data can slowwww things down.

It helps to write *optimized* code to tackle problems when you're scale becomes sufficiently large. Improving your code's speed can save you seconds, minutes or hours of twiddling your thumbs. Improving your code's memory usage can move a problem from unsolvable to solvable.

<u>Part 1:</u>
We're going to do a few whiteboard exercises today, working through simple examples to try and speed up your code. Directions will be intentionally vague sometimes… that means you need to ask questions to the instructor and/or make assumptions as appropriate!

Two rules for optimizing code today:

1. You can't optimize if your code doesn't work in the first place. So first step is successful execution.
2. You can get really serious about this[1], but that's not necessarily your comparative advantage as an economist. Use tools and tricks you know today.

<u>So your first goal at the whiteboard is come up with a solution that works. Then optimize your code at least once to improve its speed.</u>

3 tasks from more -> less complex (down from 5 for time's sake, you're welcome ☺):

1. You are given a list of *n* numbers. Return a new list of squares (^2) of all even numbers in the original list.

2. You are given an arbitrary list of integers. Return a list with no duplicates values.

3. You are given a list of integers. You are also given a single integer object, *sum = 15*. Find and return a matching pair of numbers in the list that is equal to *sum.*

<u>Part 2:</u>

If not everyone has looked yet, I wanted to spend a few minutes review some of Anton's notes in JupyterHub that we didn't have time to cover during his guest lecture. There are some very useful examples for numpy and running linear regressions…

[1] There's an inherent cost-benefit calculus that should be considered when optimizing code. How much time does it cost to improve speed/memory versus the expected gains from doing so?