Empirical Analysis of Energy Markets

Eugene Tan (TA)

9/11/2020

While you're waiting

Do the survey on courseworks!

Recitations and the course

Goals:

- understand empirical economic research and to evaluate its policy relevance (Ignacia)
- analyze real datasets and replicate empirical papers' data analysis using R (Eugene)
  - DataCamp teaches syntax, data structures
  - I'll teach everything else R related
    - These will be hands-on!
    - Come ready to code.
    - Be reasonably rude on zoom

What I'll do

1. Some programming and some basic computer science
2. Empirical Exercise prep
3. Going over Empirical Exercises
4. Applying what you do in class, estimating causal effects
5. Regression techniques for causal inference
6. Maybe some other data science stuff

R onramp (1/2)

Week	Date	TA Session	What you'll do at home
Week 1	9/11	Programming and CS Basics	Intro and "Intermediate"" R Visualization with ggplot
Week 2	9/18	Programming and Analysis Strategy, Paradigms	data.table, Tidyverse, importing csvs Excel, readr, data.table imports

R onramp (2/2)

Week	Date	TA Session	What you'll do at home
Week 3	9/25	Energy Applications	Empirical Exercise 1
Week 4	10/2	Data Science Skills	work on DS skills
Week 5	10/9	Econometrics techniques	Empirical Exercise 2

. . .

Logistics: Github

All the content I produce will be on Github.

- answer keys to empirical exercises
- examples
- these slides
- answers to any questions you have

https://github.com/taneugene/empirics_energy

For when you get stuck

- Google
  - Stackoverflow
- Raise an issue on GitHub
  - Allow me to reproduce your error
  - Show me the error, code, dataset etc. . .

About Me

- From Petaling Jaya, Malaysia
- SIPA PhD Student
  - Energy systems & energy resource economics
- Academic Background
  - CC: Environmental Chemistry (BA)
  - FES: Environmental Economics (MSc)
- Work Experience
  - Energy and Climate Geopolitics @CarnegieEndow
  - Teaching Python, Data Science, Machine Learning at DS Education Startup
  - Data Scientist at the World Bank GFDRR

This weekend on DataCamp

- Introduction to R
  - Arithmetic
  - Variables
  - characters, numerics, logicals
  - Vectors
  - Matrices
  - Factors
  - Frames
  - Lists
- “Intermediate R”
  - Conditionals
  - Loops
  - Functions
  - Apply
  - regexp, datetimes

What I'll do today

- Tie each of these things to a concept in CS
    - Computers as machines
    - Data Structures
    - Repitition
    - Abstraction
    - (maybe) Recursion
- Go over these concepts
- Type and put stuff in the console along with me!

Computers as machines

Most machines are designed to do one thing. For example, a fan is designed to blow air.

A computer is a machine:

> *"A computer does two things: perform calculation & remember the results."*

Performing Calculation - Arithmetic Operations

- Arithmetic
- Comparisons
- Relational Operators

Remembering Results - Variables

- <-
- =

Processes

With those two things, you can iterate and run a process.

A process is as a very precise sequence of steps.

Why do processes need to be precise?

(mis)interpretation

Computers are dumb!

- Humans can understand somewhat a ‘grey area’, computers needs a 1 or a 0.
- Error messages will tell you how you’re not being precise

More Definitions

- A process is a very precise sequence of steps (from the last slide)
- An algorithm is a set of steps used to solve a specific problem
    - A process is an implementation of an algorithm.
- A program is a collection of interlinked processes.
- Programming is just creating programs, which your computer is designed to run.

Remembering Results and Data Structures

- Do this in variables
- Data structures encapsulate data at once
- Different configurations of how computers store things in memory
    - Memory as an array of slots
    - Everything takes memory
    - Vectors, Matrices, Factors, Frames, Lists

Computers vs humans

- Computers are dumb, but humans are lazy
- Let computers do the work for you
- The strength of computers is repetition
  - for, while
- We can give computers some sense of logic
  - if else

Abstraction

- Low level programming is at the level of the machine (hardware)
  - Moving bits around different parts of your hard drive
  - This is your OS
- High level programming abstracts from it
  - Use processes that run other processes (that other people have worked on)
  - We do this when programming too
  - functions, libraries

Recursion (if there's time)

```r
fibonacci <- function(n){
  # Base case
  if ((n==1)|(n==2)){
    return(1)
  }
  # Recursive case
  else{
    return(fibonacci(n-1) + fibonacci(n-2))
  }
}
a <- 1:20
sapply(a, fibonacci)
```

```
##  [1]    1    1    2    3    5    8   13   21   34   55
## [16]  987 1597 2584 4181 6765
```

What we've covered

- A computer is a machine that performs computation and remembers the results
- Computers are dumb, so be precise
- A process is a list of precise step-by-step instructions (RScript)
- Data structures encapsulate data
- Humans are lazy, so we use computers to do repetitive things
- Abstraction allows you to do higher-order things
- Recursion allows repetition to have infinite power

Programming efficiently

- Test as you go! (cmd-enter on DC)
- Error messages tell you how you're not being precise enough
- Use documentation (?)
- Google and use stackoverflow a lot!

Questions

- Any for me?
- Survey results