Gov 2002 - Section 10 - The Analysis of Large-scale Data, With Applications to Diff-in-Diff Estimation

Connor Jerzak

Harvard University, Department of Government

November 16, 2016

The Analysis of Large-scale Data

Agenda:

- The theory & practice of large-scale data analysis.
- Application to differences-in-difference estimation with 1,000,000,000 observational units.

Large-scale data

Large-scale data analysis problems are becoming increasingly important. Sample of questions requiring large-scale computing resources:

- *How are demographic and voting trends related?* There are about 200,000 voting precincts in the US and over 211,000 census block groups. If we have data from 5 elections, this gives 1,000,000 observations.
- *How does voter proximity to newly built religious institutions influence their later campaign donations?* The individual-level Database on Ideology, Money, in Politics and Elections has over 100 million contribution entries covering the period from 1979 to 2012.
- *How does precinct-level unemployment in France predict vote share for the Front National?* Again, 100,000s of units.

Large-scale data

What tools do researchers need to make progress on these problems?

- Pre-compiled libraries.
- Parallelization.

Pre-compiled libraries - Background

- Default R has some drawbacks for large datasets.
  - By default, R reads data into memory.
    - Most PCs have no more than 8 GB of memory. Using more memory than available slows system to a crawl.
    - Why? Some data has to be stored in disk so thrashing occurs: data is rapidly exchanged between memory and disk, slowing down computations.
  - R is an interface to compiled code (written in C++ or Fortran), but is not as fast as programs written directly in those languages.
  - Fortunately, there are wonderful packages that greatly improve the speed of R.

Memory vs. Disk

Typical Properties	Memory	Disk
Capacity:	1-4 Gb	**>100 Gb**
When power goes off:	Data is lost	**Data is safe**
When program ends:	Data is lost	**Data is safe**
Sequential access speed:	**1.2 GB / second**	50 MB / second
Random access speed:	**1.2 GB / second**	?? ~ 66.7 seeks / second

Tradeoff:

Disks have greater capacity (more GB) and offer permanent storage;

Memory is much faster.

data.table

`data.table` can be used to manipulate datasets up to 100 GB. The speed of this package is impressive. All functions that accept that `data.frame` should work on `data.table`.

- Why is it faster?
  - `data.table` objects are read into memory like `data.frame` objects, but `data.table` operations have been optimized for speed and memory efficiency.
- Key commands:
  - `dt <- fread("data.csv")`. Use `fread` for loading in massive (or any) .csv files.
  - See `https://s3.amazonaws.com/assets.datacamp.com/img/blog/data+table+cheat+sheet.pdf` for other details about using `data.table`'s subsetting commands.

ff

`ff` allows for the analysis of even larger data - data that is far too large to store in RAM.

- How does it work?
  - Objects are stored on disk, but behavior as if they were in RAM. This works because the matrix is stored as a C++ matrix on the disk, and the `ff` package points `R` to sections of this matrix.
- Key commands:

```
library("ff")
ff_object  <- ffdf( as.ff(rnorm(100000000) )  )
ff_object
```

Parallelization

- **Goal**: Perform a series of operations on a large dataset.
  - *Example*: I have data from the French statistical agency and vote share data for 100,000s of precincts. I want to fit a series of regression models to look at changes in vote share for each of the major parties as a function of changing unemployment.
- Options: **serial** vs. **parallel** analysis. **Quantum computing** is a fear years away.

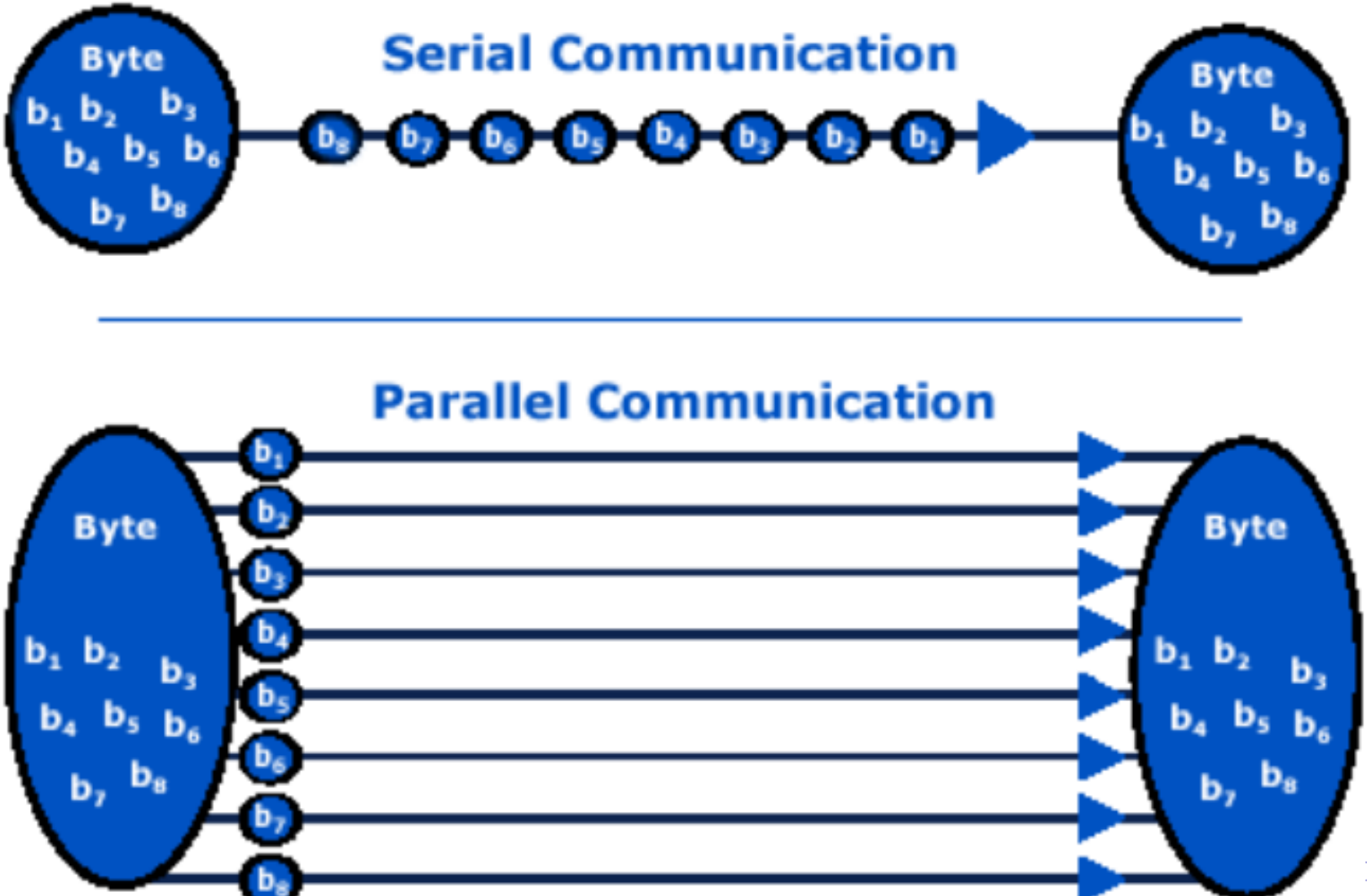
Serial Approach

- **Goal**: Perform a series of operations on a large dataset.
  - **Serial approach**: Perform operations strictly sequentially. Perform operation $i + 1$ after completing operation i.
    - Use a `for` loop to run the model for each party. Another example from teaching: 1-1 office hours.
    - Total time will be kn, where k is the amount of time it takes to perform each operation, and where n is the total number of operations.
    - *Use cases*. (1) Small datasets (e.g. many political science applications). (2) Dynamic updating required (where the sequential nature of the calculations is important; example: the outcome of one model determines the functional form of the next model).

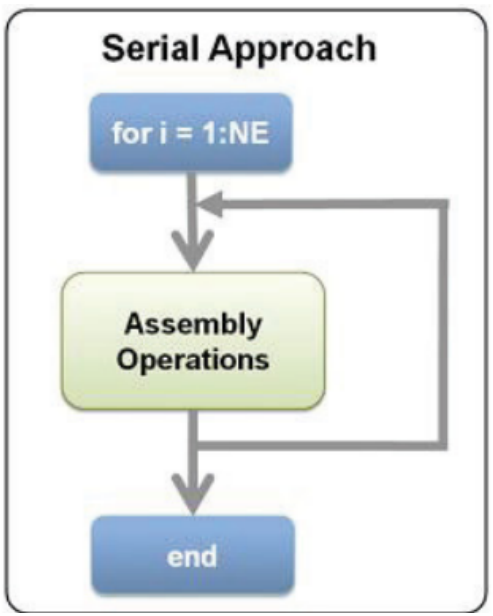
Parallel Approach

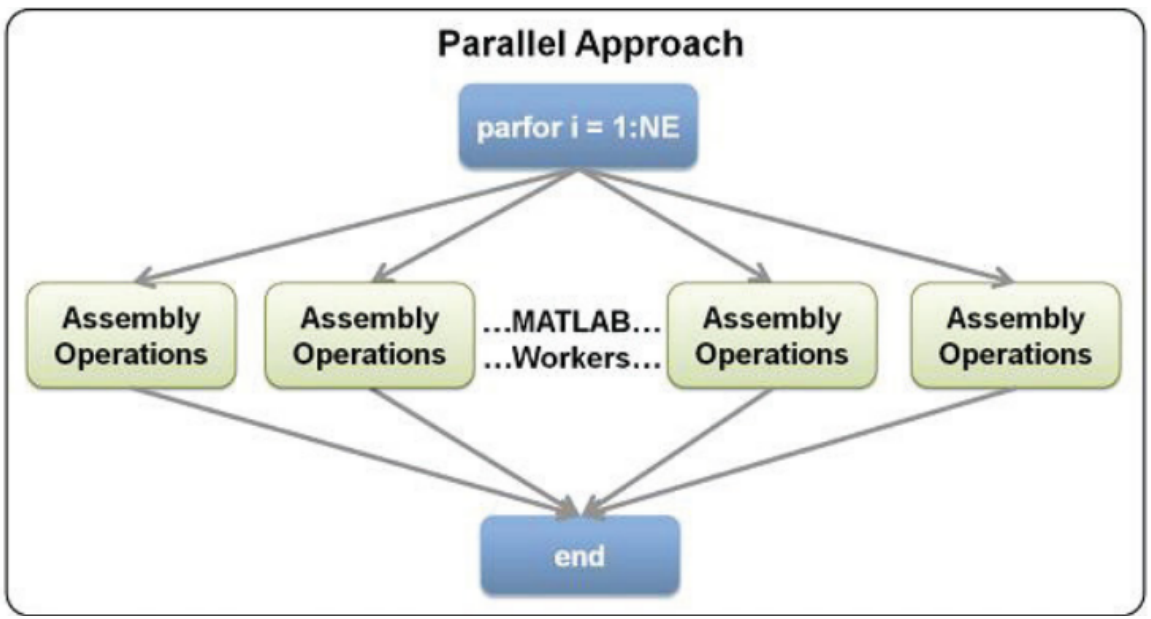
- **Goal**: Perform a series of operations on a large dataset.
  - **Parallel approach**: Perform operations simultaneously on different processors. Perform many operations at once.
    - Have each of my computer's 4 processors fit a model at once. Another example from teaching: lecturing to a class.
    - Total time will be ([very] approximately) $(kn)/p$, where k is the amount of time it takes to perform each operation, k is the number of operations, and p is the number of cores.
    - *Use cases*. (1) Analyses of large datasets where the order of each analysis is irrelevant (example: running new model for each year of data).

Parallel vs. Serial Computing



Parallel vs. Serial Computing





Parallelization in R

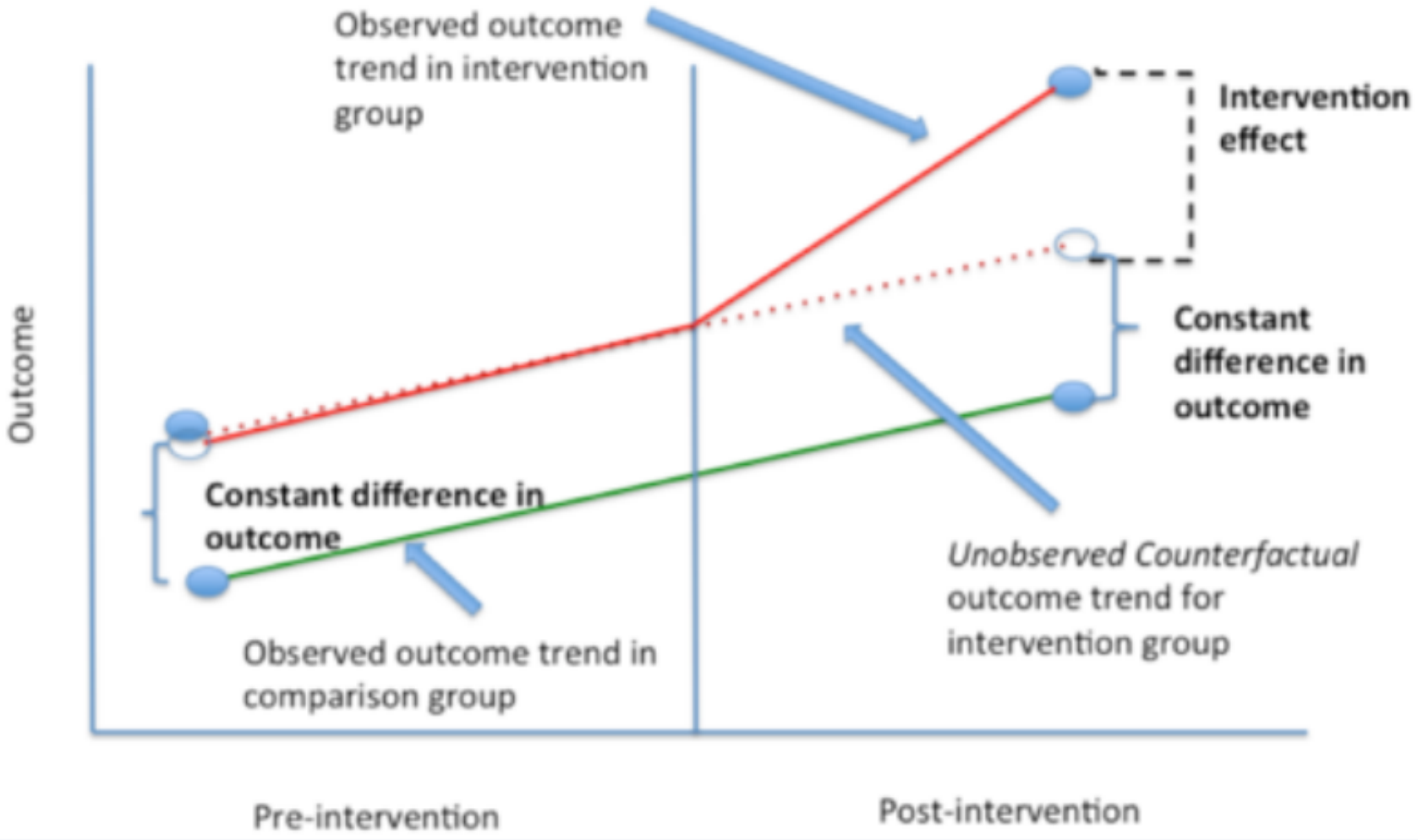
- The `foreach` and `doMC` packages are useful for researchers who want to perform parallel tasks on their PCs. Key commands: `registerDoMC`, `foreach`, `%dopar%`, and `detectCores()`.

```
#####parallel loop in R#####
library(foreach); library(doMC)
cl<-registerDoMC(4); detectCores()#use 4 cores
n_rep <- 100; x <- runif(1000000); y <- runif(1000000)
internal_loop <- foreach(i = 1:n_rep) %dopar% {
  x_reduced <- x[sample(1:length(x), 100)]
  y_reduced <- y[sample(1:length(y), 100)]
  return(  coef(lm(x_reduced ~ y_reduced))[2] )
}
coef_vec_parallel <- unlist( internal_loop )
hist( coef_vec_parallel ); cl<-registerDoMC(1)
```

Diff-in-diff - setup

- **Setup**: Diff-in-diff model used to study the effect of an in a treated and control group when randomization is not possible. It leverages over time variation to make causal comparisons.
- **Key assumption**: Parallel trends assumption: must show that over-time change in outcome for treated group *would have been the same* as the over-time change in the control group *if it weren't for the intervention*.

Diff-in-diff, visualized



Diff-in-diff - estimation

- Unconditional diff-in-diff estimator: $\hat{\delta} = (\bar{y}_{T2} - \bar{y}_{T1}) - (\bar{y}_{C2} - \bar{y}_{C1})$ where $\bar{y}_{jk}$ denotes the average of the j-th treatment group in the k-th period.
- Conditional diff-in-diff estimation: Run a linear regression, with an indicator which is 1 for post-intervention units, an indicator which is 1 for treated units, and the interaction between these two indicators. Can also condition on other covariates to improve efficiency.
  - Post-intervention indicator removes confounding due to over-time changes.
  - Treatment indicator removes baseline differences in outcomes between treated and control.
  - Interaction between the post-intervention and treatment indicator captures the intervention effect (i.e. the Δ in outcomes for the treated units in the post-intervention period, after controlling for pre-intervention differences between treated and control, as well as time trends).

The conditional regression framework:

$$Y_{i,t,k}^{\text{Obs}} = \eta \cdot 1_{T_{i,t,k}=1} + \gamma \cdot 1_{P_{i,t,k}=1} + \tau \cdot 1_{T_{i,t,k}=1, P_{i,t,k}=1} + \sum_{j=1}^{5} \beta_j \cdot X_{i,t,k,j} +$$

$$\epsilon_{i,t,k} \sim N(0, \sigma^2),$$

where

- $T_{i,t,k}$ is an indicator of whether observation i, t, k is in the treated group;
- $P_{i,t,k}$ is an indicator of whether observation i, t, k is post-intervention;
- τ is the treatment effect, and equals 2 in the simulation.

Regression with large datasets in R

- Key packages: `biglm` , `bigmemory`.
- `biglm` processes chunks of a dataset at a time, calculates the sufficient statistics required for the model, and then loads the next chunk.
- Code can be found at `http://connorjerzak.com/wp-content/uploads/2016/08/big_regression.R.zip` and on Canvas. Caution: the code will take about 5 minutes to run.

Take aways

- Donald Knuth: "The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming."
- For large-scale projects, parallelization and high performance computing can speed up publication by a factor of months.
- Use packages such as `data.table`. These are invaluable resources.

Acknowledgements

- These slides build from material presented by Pearl, Rubin, Jordan, Jackman, and others.