R-Based High Performance Computing for Social Science: Using the HMDC Cluster

MIT Political Science Methods Workshop

Soubhik Barari
Computational and Statistical Research Specialist
MIT Political Science

October 20 2017

Workshop Overview

Goals:

I. Learn the fundamentals of high-performance computing (HPC).

II. Learn how/when to use HPC to make research more efficient.

Workshop Overview

Goals:

I. Learn the fundamentals of high-performance computing (HPC).

II. Learn how/when to use HPC to make research more efficient.

Workshop Agenda:

1. Definitions and concepts
2. Tools in R
3. Using the Harvard-MIT Data Center (HMDC) Cluster
   - *Application: Determinants of civil war*
   - *Application: U.S. pairwise-jurisdiction network*

Workshop Overview

Goals:

I. Learn the fundamentals of high-performance computing (HPC).

II. Learn how/when to use HPC to make research more efficient.

Workshop Agenda:

1. Definitions and concepts
2. Tools in R
3. Using the Harvard-MIT Data Center (HMDC) Cluster
   - *Application: Determinants of civil war*
   - *Application: U.S. pairwise-jurisdiction network*

Code: `https://github.com/soubhikbarari/MITMethodsOct2017_hpc`

General Computing – Definitions

Disk:

RAM:

CPU:

General Computing – Definitions

Disk: Unit that permanently holds data for some computations to be performed (location matters).

RAM:

CPU:

General Computing – Definitions

Disk: Unit that permanently holds data for some computations to be performed (location matters).

RAM: Unit that temporarily holds data for some computations to be performed (location does not matter).

CPU:

General Computing – Definitions

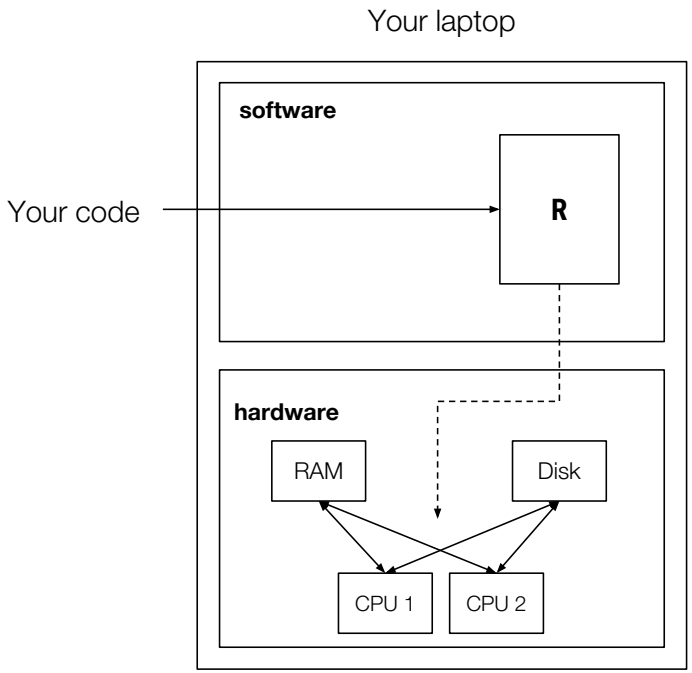
Disk: Unit that permanently holds data for some computations to be performed (location matters).

RAM: Unit that temporarily holds data for some computations to be performed (location does not matter).

CPU: Unit that performs computations, e.g. takes instructions, performs them, returns output.

General Computing – Definitions

Figure: Your machine



General Computing – Commands

How much space available on my machine? (disk)

Mac/Linux: `df -H`

Windows: `dir`

How much memory available on my machine? (RAM)

Mac/Linux: `top`

Windows: `systeminfo`

How many cores/processors on my machine? (CPU)

Mac: `sysctl -n hw.ncpu`

Linux: `nproc --all`

Windows: `systeminfo`

General Computing – Concepts

How many computations can be performed on a single CPU at a time?

General Computing – Concepts

How many computations can be performed on a single CPU at a time? Only 1.

General Computing – Concepts

How many computations can be performed on a single CPU at a time? Only 1.

Does adding more CPUs automatically speed up a script?

General Computing – Concepts

How many computations can be performed on a single CPU at a time? Only 1.

Does adding more CPUs automatically speed up a script? No. We must specify that our code's instructions need to occur over multiple CPUs.

General Computing – Concepts

How many computations can be performed on a single CPU at a time? Only 1.

Does adding more CPUs automatically speed up a script? No. We must specify that our code's instructions need to occur over multiple CPUs.

Does adding more RAM automatically speed up a script?

General Computing – Concepts

How many computations can be performed on a single CPU at a time? Only 1.

Does adding more CPUs automatically speed up a script? No. We must specify that our code's instructions need to occur over multiple CPUs.

Does adding more RAM automatically speed up a script? Only if modules and processes in your script automatically use the new RAM (most of the time, yes).

High Performance Computing – Definitions

Parallel computing:

Concurrent computing:

High performance computing:

High Performance Computing – Definitions

Parallel computing: A computing procedure where multiple computations occur simultaneously (usually on multiple CPUs).

Concurrent computing:

High performance computing:

High Performance Computing – Definitions

Parallel computing: A computing procedure where multiple computations occur simultaneously (usually on multiple CPUs).

Concurrent computing: A computing procedure where multiple computations occur 'back and forth' on one CPU.

High performance computing:

High Performance Computing – Definitions

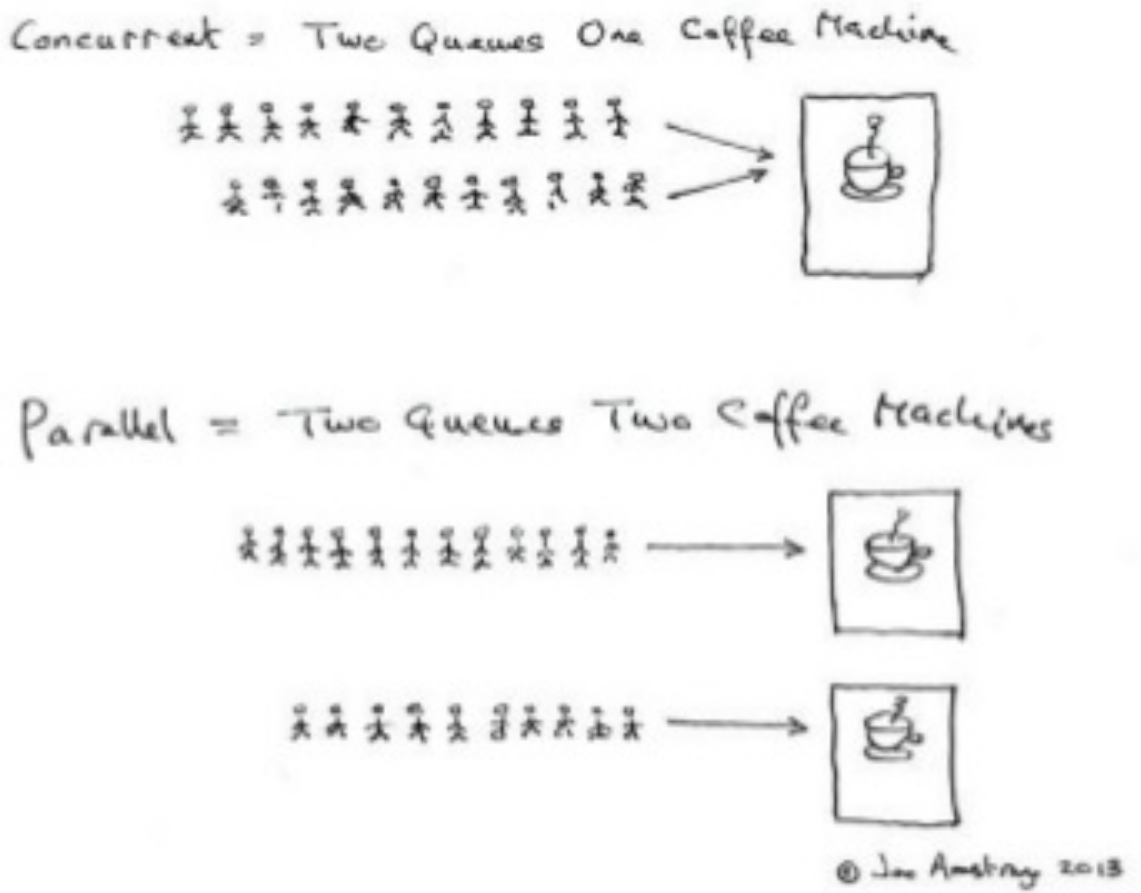
Parallel computing: A computing procedure where multiple computations occur simultaneously (usually on multiple CPUs).

Concurrent computing: A computing procedure where multiple computations occur 'back and forth' on one CPU.

High performance computing: A general software/hardware framework for performing large-scale computations efficiently (can include any combination of *parallel* and *concurrent* computing for different tasks).

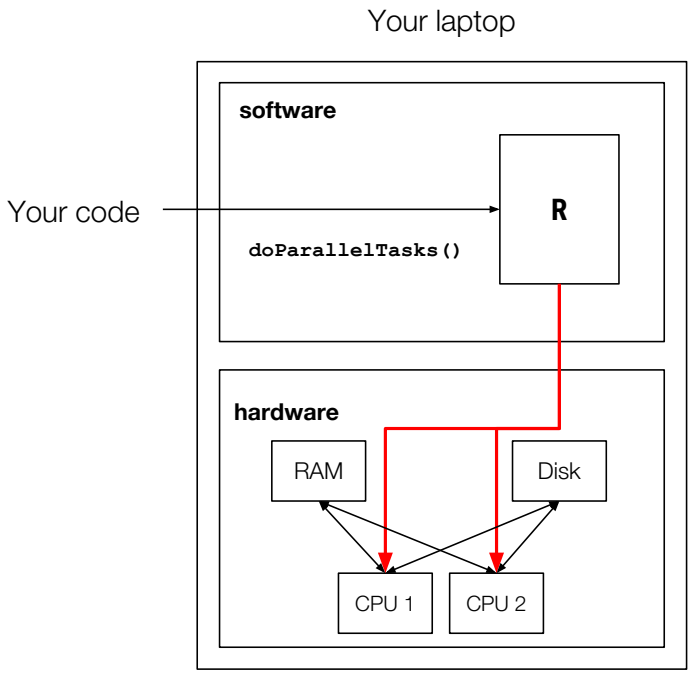
High Performance Computing – Definitions

Figure: Difference between concurrent and parallel computing



High Performance Computing – Definitions

Figure: Your machine running a script with parallelization



High Performance Computing – Concepts

What's the difference between parallel computing and high performance computing?

High Performance Computing – Concepts

What's the difference between parallel computing and high performance computing? Parallel computing is a software-level procedure. High performance computing is an overall software-hardware framework.

High Performance Computing – Concepts

What's the difference between parallel computing and high performance computing? Parallel computing is a software-level procedure. High performance computing is an overall software-hardware framework.

When should I incorporate parallel computing into my scripts?

High Performance Computing – Concepts

What's the difference between parallel computing and high performance computing? Parallel computing is a software-level procedure. High performance computing is an overall software-hardware framework.

When should I incorporate parallel computing into my scripts? As a rule of thumb, if (a.) a task takes ≥ 10 minutes to run, (b.) a loop is $\geq 1,000,000$ iterations or (c.) data is GB-scale rather than MB-scale, it's worth the *initialization cost* (see demo).

High Performance Computing – Concepts

What's the difference between parallel computing and high performance computing? Parallel computing is a software-level procedure. High performance computing is an overall software-hardware framework.

When should I incorporate parallel computing into my scripts? As a rule of thumb, if (a.) a task takes ≥ 10 minutes to run, (b.) a loop is $\geq 1,000,000$ iterations or (c.) data is GB-scale rather than MB-scale, it's worth the *initialization cost* (see demo).

When should I find a high performance computing system to work with?

High Performance Computing – Concepts

What's the difference between parallel computing and high performance computing? Parallel computing is a software-level procedure. High performance computing is an overall software-hardware framework.

When should I incorporate parallel computing into my scripts? As a rule of thumb, if (a.) a task takes ≥ 10 minutes to run, (b.) a loop is $\geq 1,000,000$ iterations or (c.) data is GB-scale rather than MB-scale, it's worth the *initialization cost* (see demo).

When should I find a high performance computing system to work with? Assess whether the costs – (a.) set-up (b.) support (c.) debugging (d.) transferring data (e.) $ – are worth the expected efficiency gains.

High Performance Computing – Concepts

A task in your script is **embarassingly parallel** if it can be split into subtasks that can be performed independently of each other.

High Performance Computing – Concepts

Is this code embarassingly parallel?

```
# Bootstrap regression
data <- read.csv("myData.csv")
mdls <- c()

for (i in 1:N) {
    mdl <- doBootstrapRegression(data)
    mdls <- c(mdl, mdls)
}

combineModels(mdls)
```

Is this code embarassingly parallel?

```
# Bootstrap regression
data <- read.csv("myData.csv")
mdls <- c()

for (i in 1:N) {
    mdl <- doBootstrapRegression(data)
    mdls <- c(mdl, mdls)
}

combineModels(mdls)
```

Yes!

Is this code embarassingly parallel?

```
# Bayesian search for optimal regression hyper-parameter
data <- read.csv("myData.csv")
prevMSE <- 0
paramSpace <- getParamSearchSpace()

for (a in paramSpace) {

    # Iteratively improve MSE until certain threshold
    MSE <- tryParameterInRegression(data, alpha=a, prev=prevMSE)
    prevMSE <- MSE

    if (MSE <= 0.001) {
        break
    }

}

mdl <- fitRegressionModel(data, alpha=a)
```

Is this code embarassingly parallel?

```
# Bayesian search for optimal regression hyper-parameter
data <- read.csv("myData.csv")
prevMSE <- 0
paramSpace <- getParamSearchSpace()

for (a in paramSpace) {

    # Iteratively improve MSE until certain threshold
    MSE <- tryParameterInRegression(data, alpha=a, prev=prevMSE)
    prevMSE <- MSE

    if (MSE <= 0.001) {
        break
    }

}

mdl <- fitRegressionModel(data, alpha=a)
```

No!

General Computing and HPC – Summary

- All computers have the same fundamental units that determine performance.
- Updates in hardware sometimes, but not always, require a change in the code we write.
- Parallel computing can improve speed if gains outweigh initialization costs.
- Using a high performance computing may be worth it for some procedures.
- Embarrassingly parallel code may be low-hanging fruit for efficiency gains.

Tools in R – Parallelization

Tools in R – Parallelization

foreach: An alternate for-loop construct that is compatible with parallelization.

```
library(foreach)

foreach(i=1:100, .combine=rbind) %do% i**2
```

Tools in R – Parallelization

foreach: An alternate for-loop construct that is compatible with parallelization.

```
library(foreach)

foreach(i=1:100, .combine=rbind) %do% i**2
```

doParallel: Provides a back-end object that allows for parallel execution through `foreach`.

```
library(foreach)
library(doParallel)

# For a single, multi-core machine
registerDoParallel(cores=2)
foreach(1:100, .combine="*") %dopar% i + (i-1)

# For a muti-node cluster
myCluster <- makeCluster(2)
registerDoParallel(myCluster)
foreach(i=1:100, .combine=cbind) %dopar% i + (i-1)
```

Tools in R – Benchmarking

Tools in R – Benchmarking

system.time: time the execution of any code bock

```
system.time(foreach(1:10, .combine="+") %dopar% i*2)
system.time(
    for (i in 1:10) {
        i*2
    }
)
system.time(myFxn())
```

Tools in R – Benchmarking

system.time: time the execution of any code bock

```
system.time(foreach(1:10, .combine="+") %dopar% i*2)
system.time(
    for (i in 1:10) {
        i*2
    }
)
system.time(myFxn())
```

microbenchmark: more accurately time execution with extra bells and whistles (e.g. repeated trials, plots, summaries)

```
library(microbenchmark); library(ggplot2)

mbm <- microbenchmark(
    for (i in 1:10) {
        i*2
    }
)
boxplot(mbm)
```

Tools in R – Exercise

1. What is the sum of the first 5 million squares of integers?

Tools in R – Exercise

1. What is the sum of the first 5 million squares of integers?

2. How long does it take for a sequential program in **R** to find the answer?

Tools in R – Exercise

1. What is the sum of the first 5 million squares of integers?

2. How long does it take for a sequential program in **R** to find the answer?

3. How long does it take for a parallelized program in **R** to find the answer?

Tools in R – Exercise

1. What is the sum of the first 5 million squares of integers?

2. How long does it take for a sequential program in **R** to find the answer?

3. How long does it take for a parallelized program in **R** to find the answer?

4. Are the results what you expected?

Tools in R – Misc.

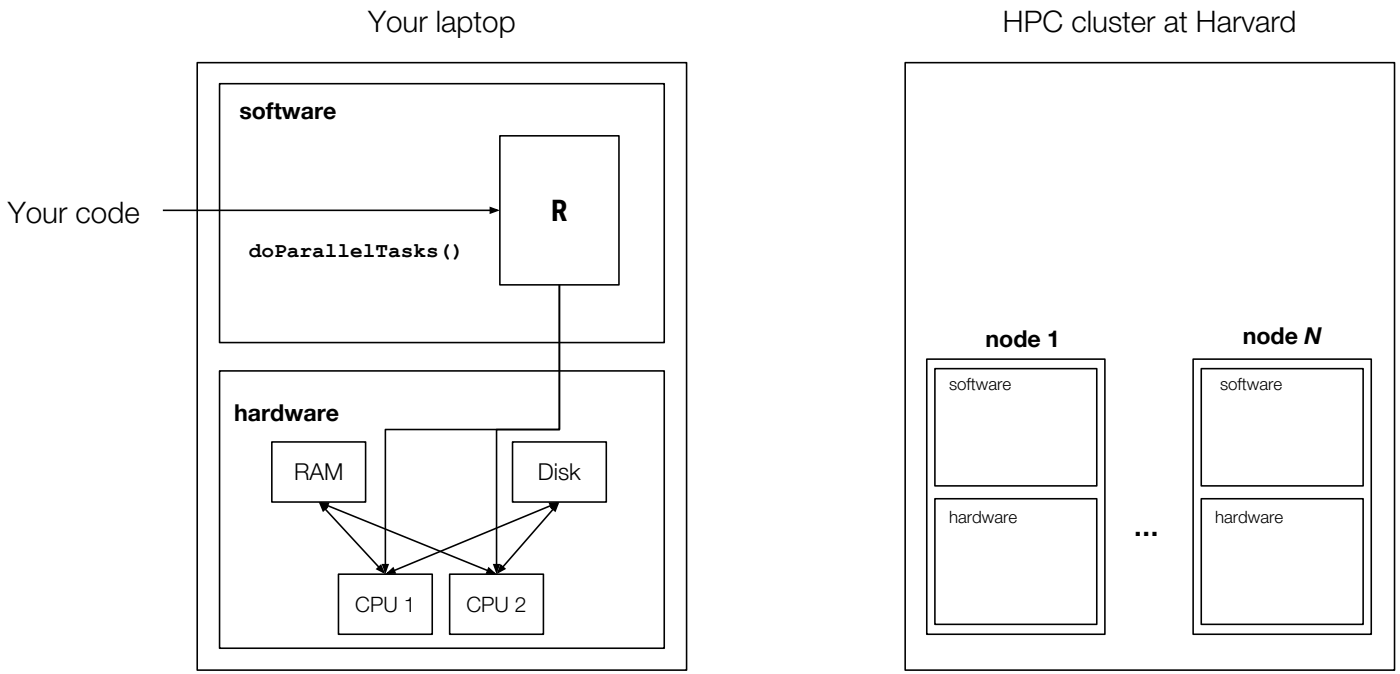
- **`mclapply`**: `lapply`-styled parallelism that uses shared memory between tasks.
- **`multidplyr`**: distribute `dplyr` operations over cores.
- **`Rcpp`**: speed up complex R operations (e.g. linear algebra) by re-writing in underlying C++.

Using the HMDC Cluster

- The HMDC cluster is free for all Harvard/MIT researchers.
- See workshop instructions sheet for account request and set-up.

Using the HMDC Cluster

Figure: The HMDC cluster



Using the HMDC Cluster – Definitions

Operating system (OS):

File system (FS):

Command line interface (CLI):

Using the HMDC Cluster – Definitions

Operating system (OS): Process that manages software tasks (your code) and hardware resources (CPU, RAM, disk).

File system (FS):

Command line interface (CLI):

Using the HMDC Cluster – Definitions

Operating system (OS): Process that manages software tasks (your code) and hardware resources (CPU, RAM, disk).

File system (FS): Part of the operating system that manages disk (your Finder is an interface to this).

Command line interface (CLI):

Using the HMDC Cluster – Definitions

Operating system (OS): Process that manages software tasks (your code) and hardware resources (CPU, RAM, disk).

File system (FS): Part of the operating system that manages disk (your Finder is an interface to this).

Command line interface (CLI): 'Bare-bones' universal program to execute other programs, send commands OS, or to navigate FS (your Desktop is a substitute for this).

Using the HMDC Cluster – Definitions (cont'd)

Virtual Machine (VM):

`ssh`:

Using the HMDC Cluster – Definitions (cont'd)

Virtual Machine (VM): A software-based emulation of a computer/cluster (may not have an actual hardware box).

`ssh`:

Using the HMDC Cluster – Definitions (cont'd)

Virtual Machine (VM): A software-based emulation of a computer/cluster (may not have an actual hardware box).

`ssh`: A program to access a remote computer.

Using the HMDC Cluster – Specs

Space (disk)

- **500 MB** account space per user.
- **1 TB** space shared by all users.

Running scripts

(a.) *Interactive jobs cluster*

(b.) *Batch jobs cluster*

Using the HMDC Cluster – Specs

Space (disk)

- **500 MB** account space per user.
- **1 TB** space shared by all users.

Running scripts

(a.) *Interactive jobs cluster*

- Access using web client
- Real-time
- **8** nodes
- **12 CPU** per node
- **250 GB RAM** per node

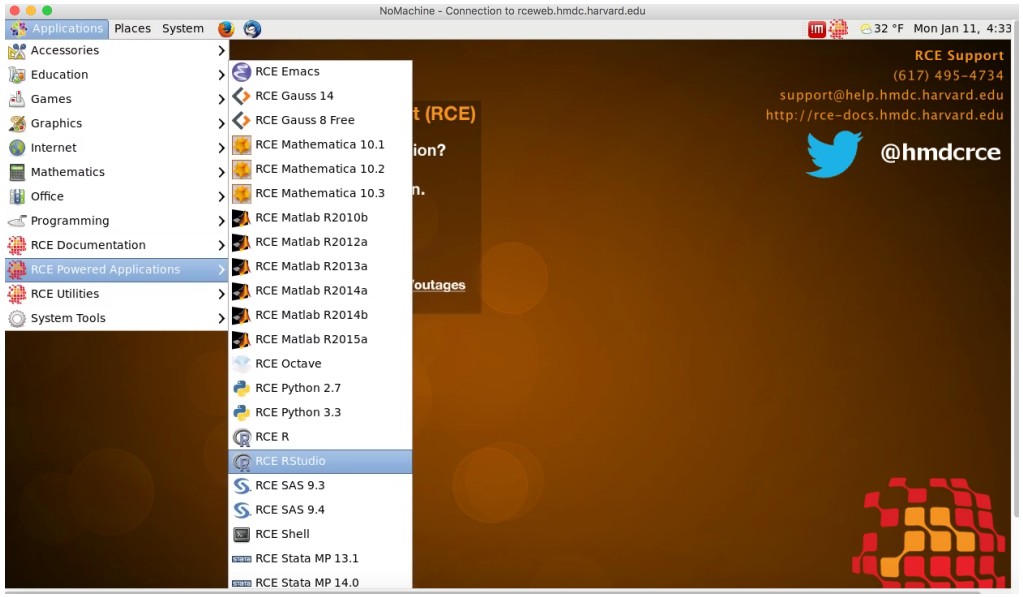
(b.) *Batch jobs cluster*

- Access using **`ssh`**
- Asynchronous
- **5** nodes
- **16 CPUs** per node
- **125GB RAM** per node

Using the HMDC Cluster – Access

Option A: Web client[1]

1. Use the **NoMachine** web client to log into your RCE account.
2. Deploy an interactive job on the desktop of the VM.

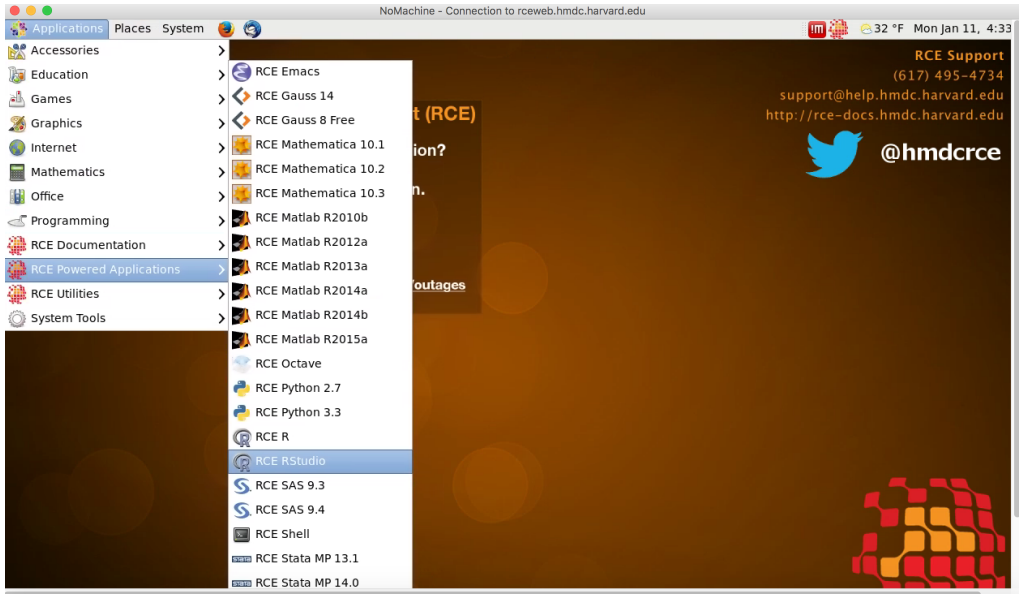


[1] See instructions sheet to download and install.

Using the HMDC Cluster – Access

Option A: Web client[1]

1. Use the **NoMachine** web client to log into your RCE account.
2. Deploy an interactive job on the desktop of the VM.



Pro: run your script in real time *Con:* limited to only 24 cores

[1] See instructions sheet to download and install.

Using the HMDC Cluster – Access

Option B: `ssh` [2]

1. Write a `.submit` file for **Condor job tracker** describing your job (see next slide).
2. Copy over `.submit` and `.R` files of job using **`scp`**.
3. Log into account using **`ssh`**.
4. Deploy batch job to **Condor** job tracker using **`condor_submit`**.
5. Track job using **`condor_status`** *or* viewing the resulting log file.

Example:

```
cd </path/to/my/files>
scp <myJob>.submit <myAccount>@rce.hmdc.harvard.edu
scp <myScript>.R <myAccount>@rce.hmdc.harvard.edu
ssh <myAccount>@rce.hmdc.harvard.edu
condor_submit <myJob>.submit
condor_status
```

[2] See instructions sheet on using command line.

Using the HMDC Cluster – Access

Option B: `ssh` [2]

1. Write a `.submit` file for **Condor job tracker** describing your job (see next slide).
2. Copy over `.submit` and `.R` files of job using **`scp`**.
3. Log into account using **`ssh`**.
4. Deploy batch job to **Condor** job tracker using **`condor_submit`**.
5. Track job using **`condor_status`** *or* viewing the resulting log file.

Example:

```
cd </path/to/my/files>
scp <myJob>.submit <myAccount>@rce.hmdc.harvard.edu
scp <myScript>.R <myAccount>@rce.hmdc.harvard.edu
ssh <myAccount>@rce.hmdc.harvard.edu
condor_submit <myJob>.submit
condor_status
```

Pro: no resource limits

Con: job may wait in queue

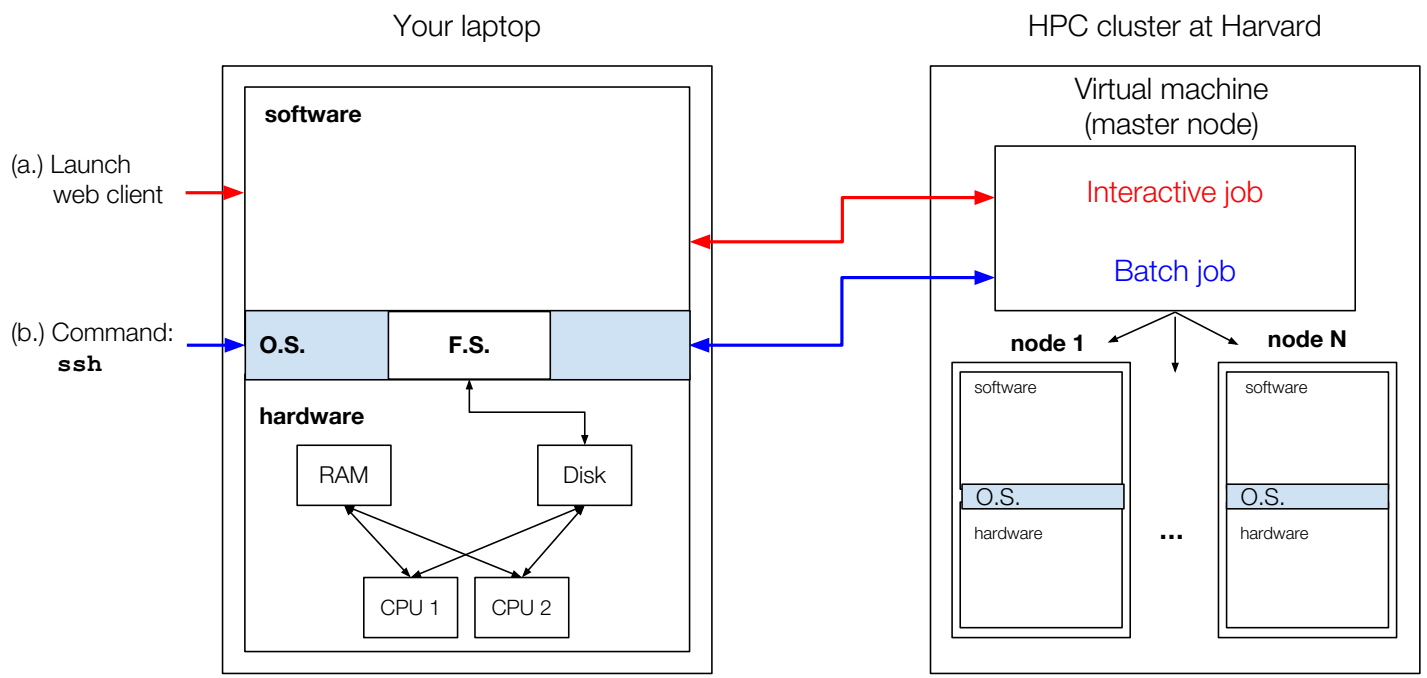
[2] See instructions sheet on using command line.

Sample Condor job submit file (`myJob.submit`)

```
# Job execution
Universe         = vanilla
Executable       = /usr/local/bin/R
Arguments        = --no-save --no-restore

# Requests
request_cpus     = 10
request_memory   = 100 MB

# Job scripts
input            = ./<myScript>.R
output           = ./output.txt
error            = ./error.txt
Log              = ./log.txt

# How many runs?
Queue            1
```

Using the HMDC Cluster – Access Overview

Figure: Your machine using the HMDC cluster



O.S. = operating system
F.S. = file-system

Using the HMDC Cluster – Applications

1. **What are the determinants of civil war?**
2. **What are the pairwise distances between U.S. counties?**

code: `https://github.com/soubhikbarari/MITMethodsOct2017_hpc`

Using the HMDC Cluster – Application (1)

What are the determinants of civil war?

Using the HMDC Cluster – Application (1)

What are the determinants of civil war?

- "Ethnicity, Insurgency, and Civil War" (Fearon et al. 2003)
- Most cited contemporary article in comparative politics (6847)
- Replication data (`repdata.zip`) at `https://web.stanford.edu/group/ethnic/publicdata/publicdata.html`

Using the HMDC Cluster – Application (1)

What are the determinants of civil war?

Bootstrap regression of civil war onset on system/dyadic variables:

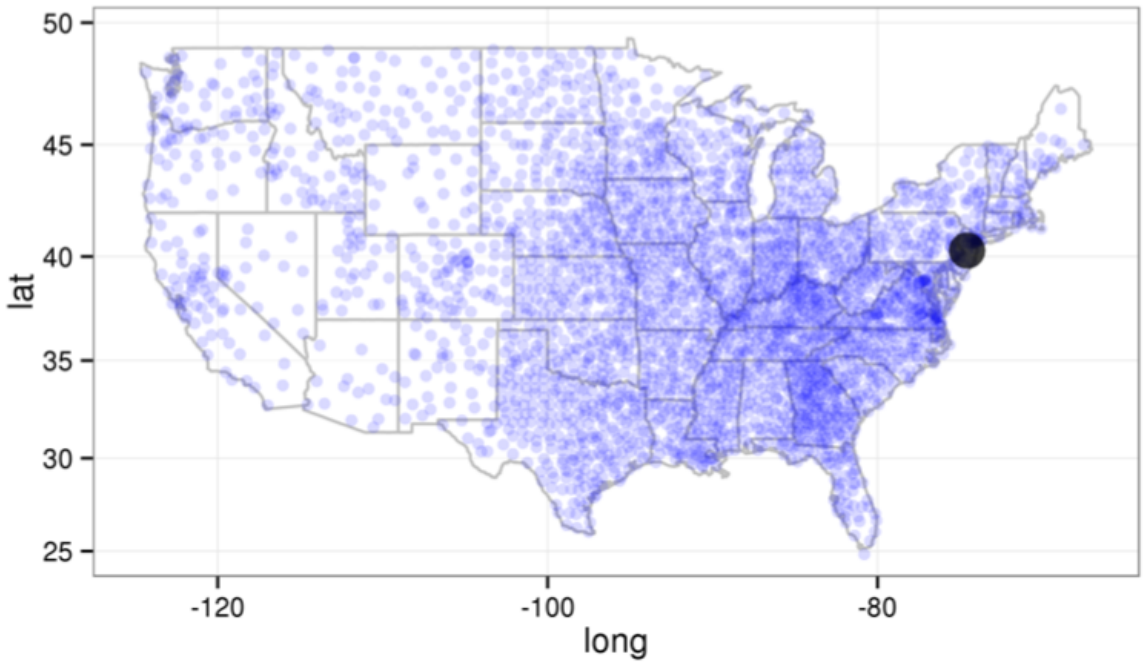
```
library(haven)
library(MASS)
library(foreach)
library(doParallel)

d <- read_dta("repdata.dta")
d <- d[d$onset != 4,]
formula <- onset ~ warl + gdpenl + lpopl1 + lmtnest + ncontig + Oil +
                   nwstate + instab + polity2l + ethfrac + relfrac

dobootstrap <- function(...) {
    # Create a bootstrap sample
    idxs <- sample(1:nrow(d), replace=TRUE)

    # Create a bootstrap sample
    D <- d[idxs,]

    # Get logit estimates
    fit <- glm(formula, data=D, family=binomial(link = "logit"))
}
```

Using the HMDC Cluster – Application (1)

Lesson: *User time* does not gain efficiency from moving to HPC, but does *system time* can. Test this out using small-n samples. Check for efficiency gains by upping sample size.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

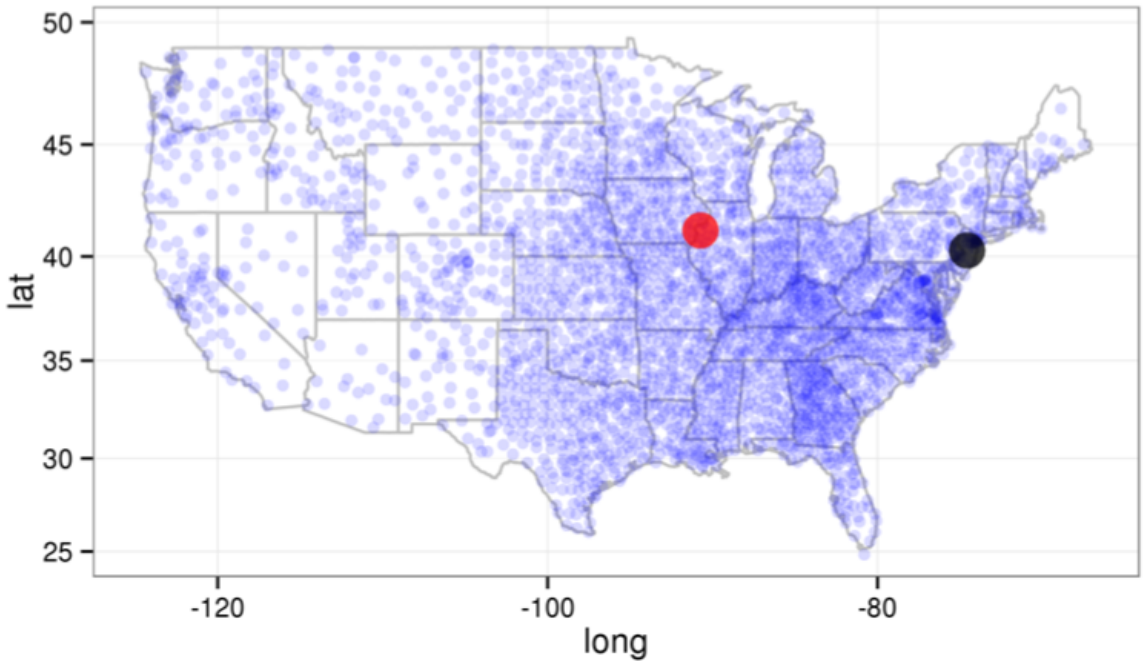


- Start with a county Mercer County, NJ

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

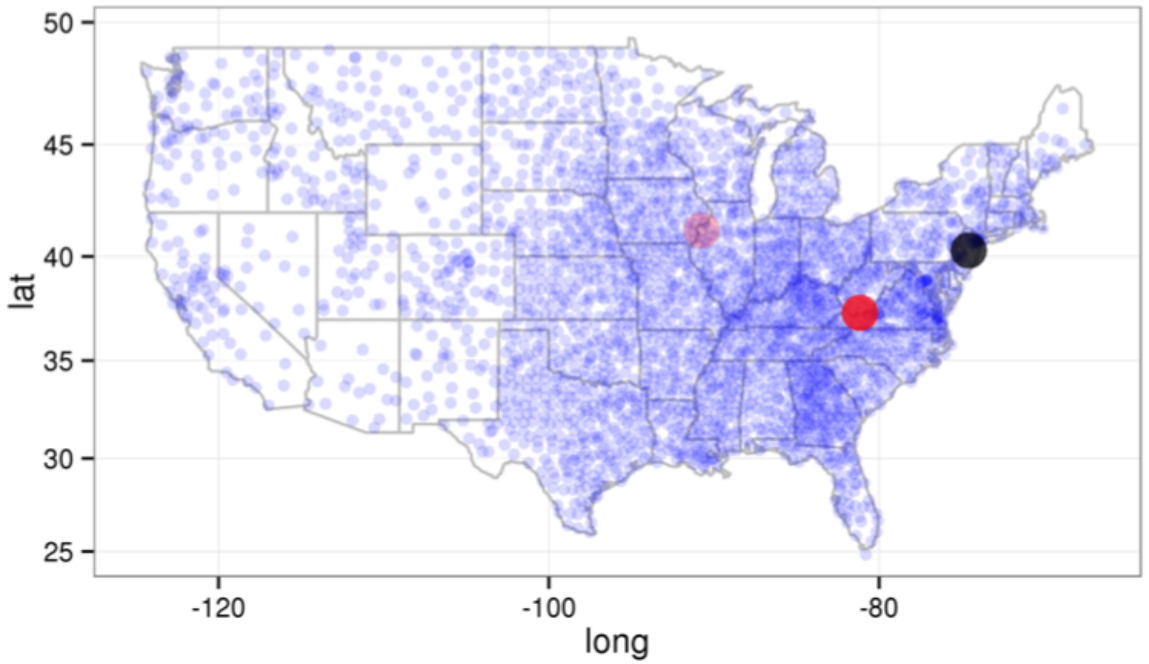


- Calculate the distance between Mercer County, NJ and Mercer County, IL.
- Store it.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

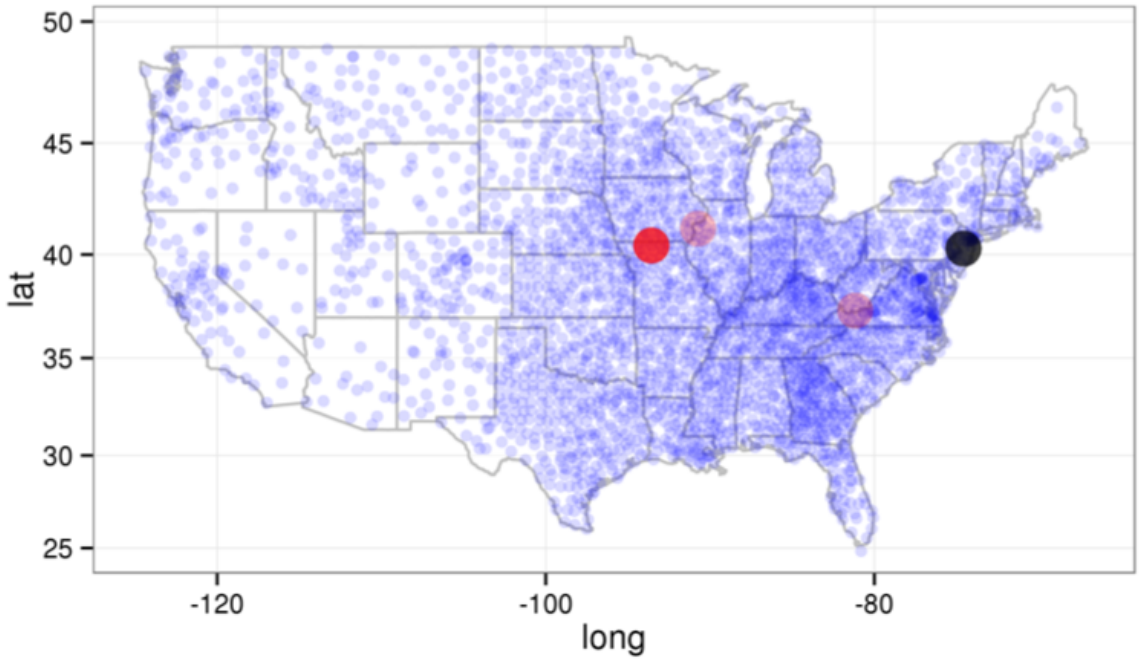


- Then, calculate the distance between Mercer County, NJ and Mercer County, WV.
- Store it.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

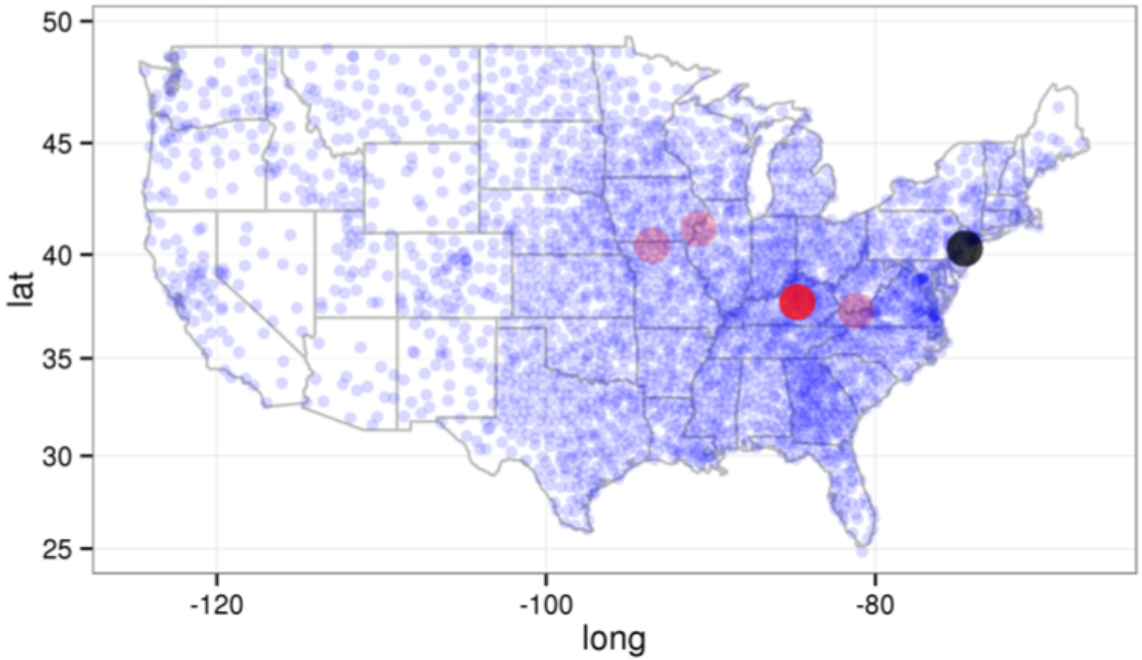


- Then, Mercer County, MO and store it.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

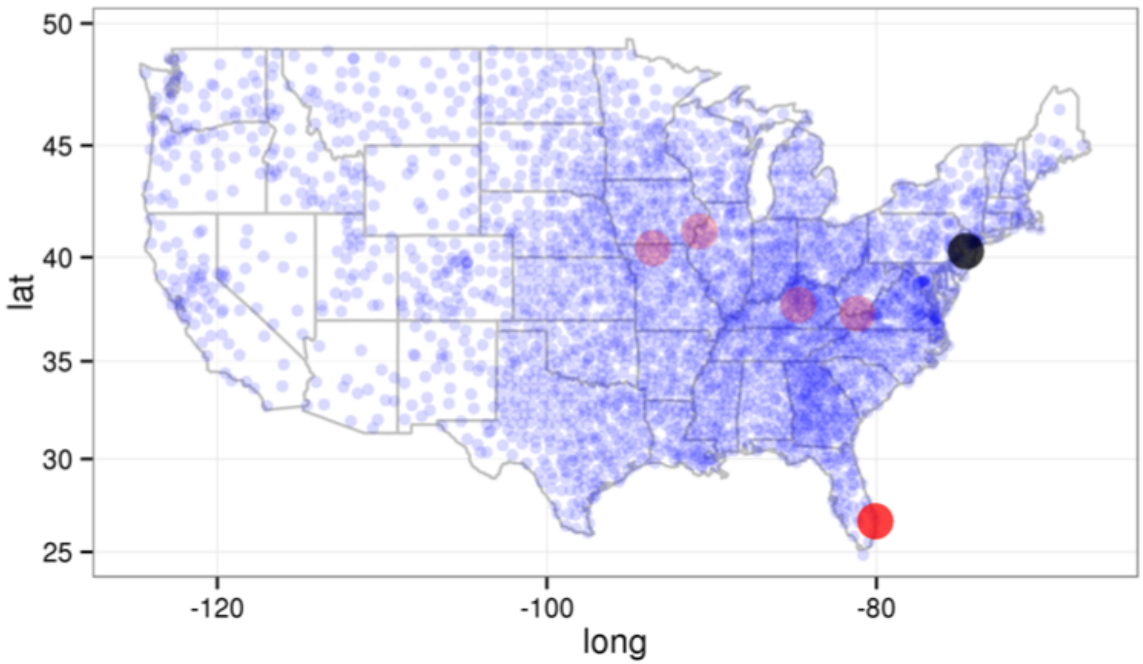


- Then, Mercer County, KY and store it.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

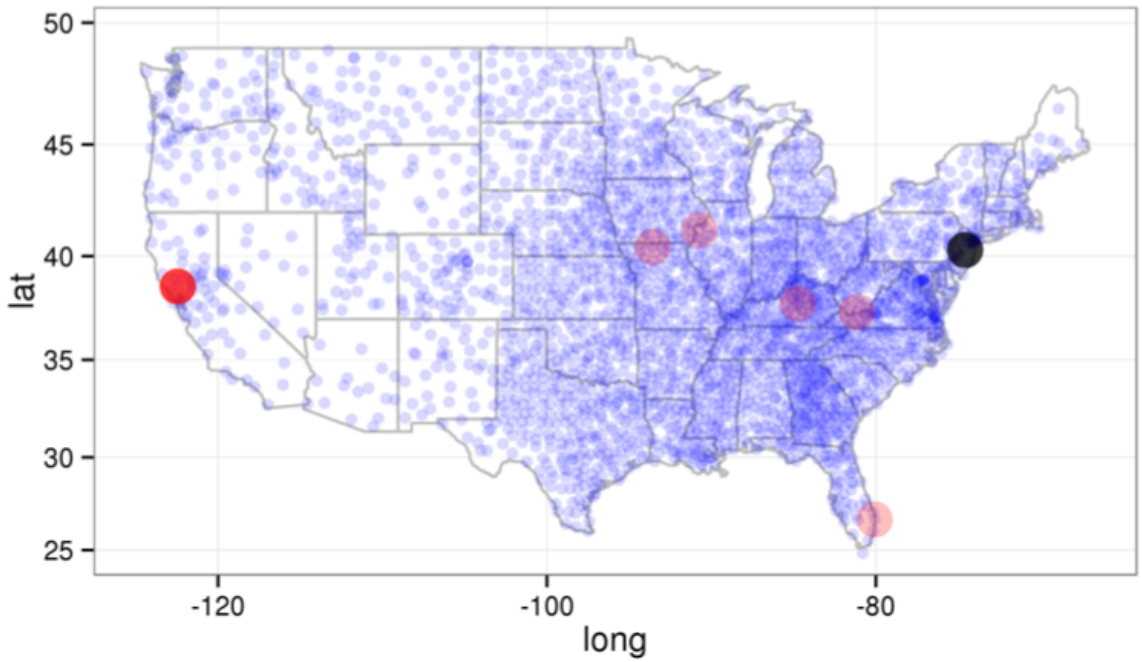


- Eventually, you would calculate the distance between Mercer County, NJ and Naples County, FL.
- Then store that.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

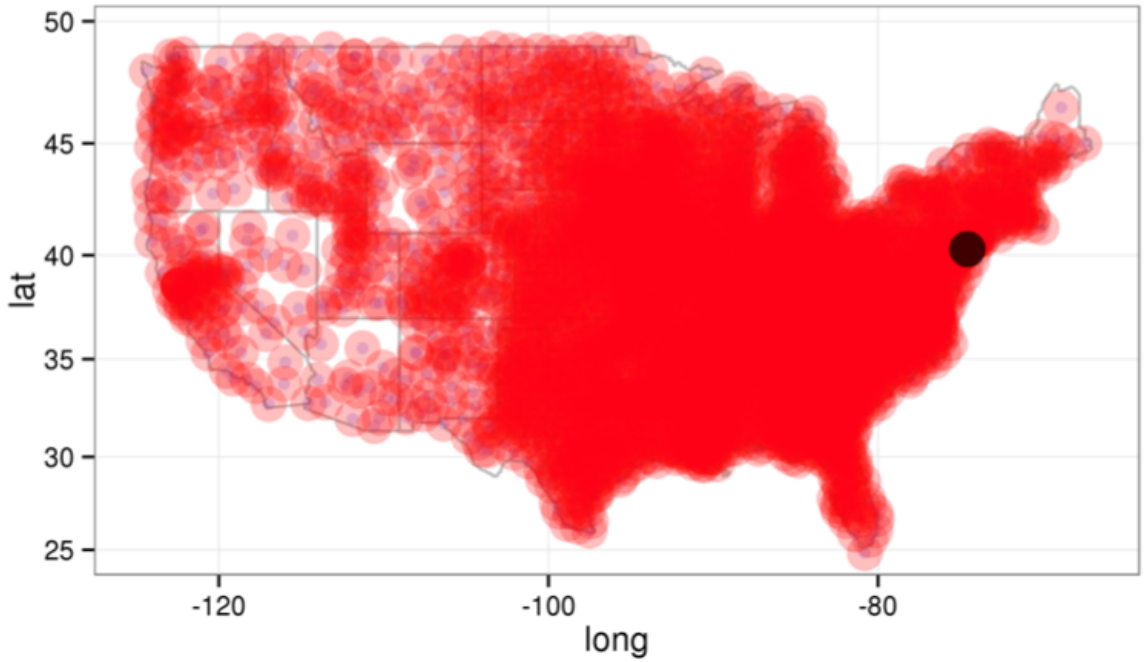


- And, then, you would calculate the distance between Mercer County, NJ and Napa County, CA.
- Then store that.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]



- For Mercer County alone, thats 3,108 distance calculations.
- Without duplicate calculations: $\frac{3{,}109 \cdot 3{,}108}{2} = 4,831,386$ distances.
- With duplicate calculations: $9,665,881$ distances.

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

Perform computation by element:

```
dfCounties <- read.csv("counties.csv")
dfCounties <- na.omit(dfCounties)

mCounties <- as.matrix(dfCounties[, 1:2])
mCountiesSmall <- mCounties[1:400, ]

calcPWDe <- function(mat) {
    # Brute force calculation of each individual pair
    out <- matrix(data = NA, nrow = nrow(mat), ncol = nrow(mat))

    for (row in 1:nrow(out)) {
        for (col in 1:ncol(out)) {
            # Calculate Euclidean distance
            out[row, col] <- sqrt(((mat[row, 1] - mat[col, 1]) ^ 2 +
                                   (mat[row, 2] - mat[col, 2]) ^ 2))
        }
    }
    return(out)
}
```

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

What are the pairwise distances between U.S. counties?[3]

Now perform computation by row:

```
dfCounties <- read.csv("counties.csv")
dfCounties <- na.omit(dfCounties)

mCounties <- as.matrix(dfCounties[, 1:2])
mCountiesSmall <- mCounties[1:400, ]

calcPWDv <- function(mat) {
    # Distance calculation over rows
    out <- matrix(data = NA, nrow = nrow(mat), ncol = nrow(mat))

    for (row in 1:nrow(out)) {
        out[row, ] <- sqrt(((mat[row, 1] - mat[, 1]) ^ 2 +
                            ##                !^!
                            (mat[row, 2] - mat[, 2]) ^ 2
                            ##                !^!
                            ))
    }
    return(out)
}
```

[3] Adapted from Princeton ASPC 2015 workshop.

Using the HMDC Cluster – Application (2)

Lesson: Optimize your code for obvious gains before doing any parallelization or HPC. This might also involve modifying code to better split up tasks between cores.

Workshop Summary

Takeaways:

- Understand why and how parallelization improves perfomance.
- Know how to implmenet parallelization in R both locally and on cluster.
- Know when parallel and HPC aren't worth it.

Further applications:

- Parametric bootstrap
- Cross validation
- Markov chain Monte Carlo
- Bayesian estimation

Resources

- **Research Computing Environment (RCE):**
  `http://rce-docs.hmdc.harvard.edu/book/accessing-rce-0`
- **Parallelization in R:**
  `https://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf`
- **Condor job tracker:**
  `http://research.cs.wisc.edu/htcondor/manual/v7.6/ref.html`
- **Build your own virtual machhine:**
  `https://aws.amazon.com/`
- **General research consultation:**
  Feel free to contact me at `sbarari@mit.edu`

Coming soon: MIT Political Science's very own HPC machine (`xvii`)