Problem set 4 - CSCI E-88b Spring 2020

Due date: Friday, May 8 2020, 10pm EST.

Please write down the last 5 digits of your Harvard ID: 70196

Please write down the URL of your Renku project: https://renkulab.io/projects/stephen.t.knapp14/problem4 (https://renkulab.io/projects/stephen.t.knapp14/problem4)

Please indicate who you may have worked with on this problem and which section you worked with someone on. If you did not work with anyone then you can leave this blank: xxxxx

This problem set corresponds to 1/6 of your final grade. It is graded out of 10.

Question 4.0: 0pt

Question 4.1: 2pts

Question 4.2: 2pts

Question 4.3: 4pts

Question 4.4: 2pts

Question 4.0: Keep it private!

If your problem set is not private, your grade will be 0/10.

Question: Please confirm that your project is private. You can edit the markdown cell from `[ ]` to `[x]` to check a box.

Your answer (4.0):

- [X] Yes, my project is private.
- [ ] No, my project is not private.

Objectives

This problem set aims at giving you a sense of how we can use create HPC workflows using several big data tools. We are going to use the Harvard Cannon cluster to geo-process large amounts of data.

Question 4.1: Interpret geo-processing code

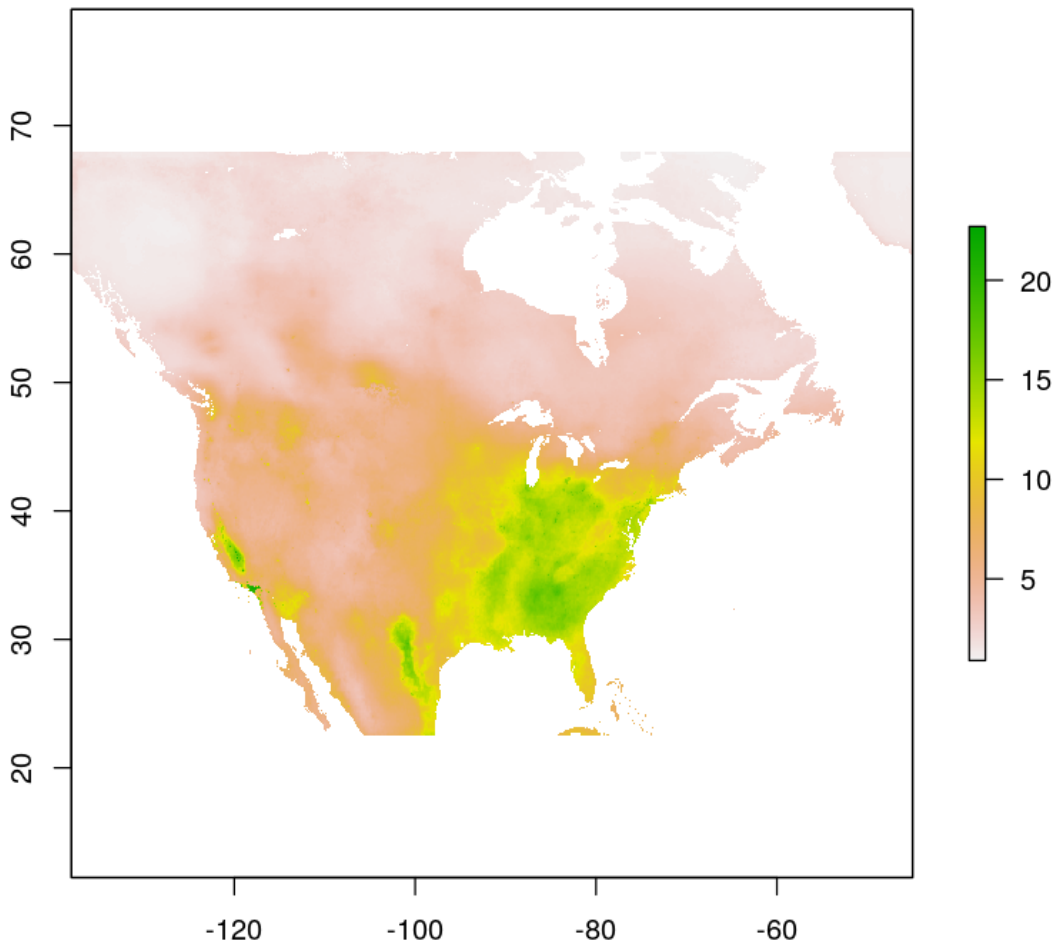
We want to use the North American Regional Estimates of PM2.5 from http://fizz.phys.dal.ca/~atmos/martin/?page_id=140 (http://fizz.phys.dal.ca/~atmos/martin/?page_id=140). Annual data is available for years 2000-2018 at the grid level.

In [1]:
```
library(raster)
library(maptools)
library(proj4)
library(ncdf4)
```

```
Loading required package: sp
Checking rgeos availability: TRUE
```

In [2]:
```
grid_file <- "../data/pm25/pm2000.nc"
grids <- raster(grid_file)
```

In [3]:
```
plot(grids) # just to get a sense of the data; not needed in Question 4.2
```



In [4]:
```
poly_file <- "../data/tl_2010_us_state00/tl_2010_us_state00.shp"
poly <- shapefile(poly_file)
```

In [5]:
```
proj4string(grids) <- proj4string(poly)
```

This step is going to require time (~15 min):

In [6]:
```
ex <- extract(grids, poly, fun = mean, na.rm = TRUE, df = TRUE)
```

In [7]:
```
result <- data.frame(ex)
result$State <- poly@data$NAME00
result$ID <- NULL
```

In [8]:
```
names(result) <- c("PM25_2000","State")
result
```

A data.frame: 52 × 2

PM25_2000	State
<dbl>	<chr>
9.072290	New York
12.341906	West Virginia
10.025059	Rhode Island
NA	Puerto Rico
8.188588	Michigan
13.711564	Tennessee
5.822106	North Dakota
7.733470	California
5.304781	Nevada
15.240623	Georgia
6.257286	South Dakota
5.479989	Utah
14.082421	South Carolina
1.752601	Alaska
12.959899	Mississippi
12.682241	Arkansas
10.511416	Missouri
11.760442	Pennsylvania
9.926818	Iowa
8.320212	Kansas
5.744527	Colorado
11.752861	Florida
12.822392	New Jersey
13.860695	Delaware
5.684786	Montana
6.558850	Washington
10.777092	Connecticut
14.194189	Indiana
9.570973	Massachusetts
12.485055	Louisiana
15.361901	Alabama
13.452344	North Carolina
13.507160	Maryland
13.660611	Kentucky

PM25_2000	State
<dbl>	<chr>
6.416755	Idaho
5.093982	Wyoming
7.169713	Nebraska
NA	Hawaii
7.240149	New Hampshire
5.278993	Oregon
13.998440	Ohio
9.213114	Oklahoma
6.282766	Arizona
8.765978	Wisconsin
7.363702	Minnesota
14.925137	District of Columbia
12.496229	Virginia
9.433195	Texas
5.443897	New Mexico
7.120827	Vermont
5.385208	Maine
12.809383	Illinois

Question (4.1): What are the objectives of the code chunks of Question 4.1? (Please limit your answer to one or two sentences.)

Your answer (4.1): Chunk 1: This chunk loads several libraries into our working images and makes availble the function in those libraries/packages for us to use that do not exist in the base R. Raster is for, "reading, writing, manipulating, analyzing and modeling of gridded spatial data." It can also process large datasets. Maptools provides a "set of tools for manipulating geographic data." Proj4, "allows transformation of geographic coordinates from one projection and/or datum to another." Finally, ncdf4 is for utilizing netCDF files, "which are binary data files that are portable across platforms and include metadata information in addition to the data sets."

Chunk 2: The first line assigns the file path for a datafile to the variable "grid_file". The second line uses the raster command on the grid_file variable which creates a RasterLayer object that with parameters that match the .nc file but MAY or MAY NOT include any actual data. The RasterLayer object was assigned the name "grids"

Chunk 3: The plot command is used on the "grids" object to generate a heatmap of North America. Now we have a better idea of the data we are working with and we know that the RasterLayer created in chunk 2 does indeed include pixel data that was used to generate the heatmap.

Chunk 4: The first line assigns a shape (.shp) file's path to the object named, "poly_file." The second line uses the shapefile command to read in the poly_file object which contains spatial data. Shapefiles consist of, at a minimum, four sub-files that contains data on attributes, geometry, index and the coordinate reference system. This particular shape file contains 5 files.

Chunk 5: The proj4string() function is used on the poly object to identify its coordinate reference systems. The identified coordinate reference systems is then applied as the same coordinate reference system for the grids object. Functionally, this allows the data in grids to be partitioned into US states.

Chunk 6: The extract command takes the data in the grids object and provides the mean values of those data in the groupings defined in the US state-shaped polygons contained in the poly object. If the function finds any NA values it will remove them before the calculation. The output of the means by state will be in a R dataframe. This dataframe is assigned the name "ex."

Chunk 7: The first line takes the object "ex" which is already a dataframe (because df=TRUE for the last command in chunk 6) and turns it into a dataframe again named "result". The second line takes the names of the US states contained in poly object, creates a new column on the results dataframe and inserts those state names that correspond to the means. The third line remove an index column in the results dataframe that is no longer needed now that the states names are in the dataframe.

Chunk 8: The first line names the columns of the results dataframe. The second line prints the dataframe "results".

Question 4.2: Upload data to Harvard Cannon

Question (4.2): We want to run the compuations of Question 4.1 on the cluster. PM2.5 data is already available on Harvard Cannon, but the shapefile is not. So the first step is to copy folder `data/tl_2010_us_state00` to your home folder (or a folder of your choice in home home folder) on the cluster. Describe the steps and commands that we are using.

Your answer (4.2):

Step 1: Copy contents of folder data/tl_2010_us_stat00 to the cluster home folder using the following line of code entered into a Renku terminal (not on the Harvard Cannon terminal):

$ scp -r ./data/tl_2010_us_state00 e88bu2044@login.rc.fas.harvard.edu:~/

Breakdown of the command:

```
scp: is the secure copy command

-r: is the recursive switch that let's the command know I want to copy everything i
n the "from" directory

./data/tl_2010_us_state00: is the location of the directory folder on Renku that I
 want to copy the contents of

e88bu2044@login.rc.fas.harvard.edu:~/: tells the command I want to take the copied
 file and put it in my home directory on Harvard Cannon
```

Step 2: confirm the files have been copies to my home on Cannon:

(on a terminal on the Cannon log-in node): $ ls

```
Documents  singularity  tl_2010_us_state00
```

This shows the directory has been successfully copied

Question 4.3: Run geo-processing code at scale on Harvard Cannon

Run the geo-processing for years 2000-2018 as a Harvard Cannon `sbatch` job.

PM2.5 data is available in folder `/n/holyscratch01/e88b/problem4/data/pm25` .

Use `singularity` with Rocker geospatial and gather the geo-processing results for years 2000-2018 as a CSV file `result_linkage.csv` with columns `State` , `PM25_2000` , ..., `PM25_2018` , sorted in alphabetical order of `State` .

Question (4.3): Please provide all the `sbatch` -related material you submitted.

Your answer (4.3):

Step 1: I wrote 4 scripts (2 .batch and .R pairs). The first pair is for extracting the data for each year. The second pair is for aggregating the data into 1 .csv file.

problem4.batch file:

```
#!/bin/bash
#SBATCH -p shared
#SBATCH --mem 8g
#SBATCH -t 0-6:00
#SBATCH -c 5
#SBATCH -a 2000-2018      # Array of 19 jobs, with ids 1, 2, ...19
#SBATCH -o out_%A_%a.out  # starndard output
#SBATCH -e err_%A_%a.out  # standard error

## EXECUTE CODE ##
singularity exec geospatial_latest.sif R CMD BATCH problem4rcode.R script_$SLURM_AR
RAY_TASK_ID

/// END OF FILE ///
```

problem4rcode.R file:

```
#load required libraries
library(raster)
library(maptools)
library(proj4)
library(ncdf4)

#define the function to analyze each year
one.year <- function(year){

    #create grid from data
    grid_file <- paste0("/n/holyscratch01/e88b/problem4/data/pm25/pm",year,".nc")
    grids <- raster(grid_file)

    #use shape file to overlay state boundaries
    poly_file <- "~/tl_2010_us_state00/tl_2010_us_state00.shp"
    poly <- shapefile(poly_file)

    #map state boundries to grid data
    proj4string(grids) <- proj4string(poly)

    #take means by state
    ex <- extract(grids, poly, fun = mean, na.rm = TRUE, df = TRUE)

    #convert means to a data frame and add state names from the shape file
    result <- data.frame(ex)
    result$State <- poly@data$NAME00

    #drop the ID column and assign column headers
    result$ID <- NULL
    names(result) <- c(paste0("PM25_",year), "State")
```

```
    #function returns the analyzed data by state in a data frame
    return(result)
}

#set the year for this run
id <- as.numeric(Sys.getenv("SLURM_ARRAY_TASK_ID"))

#run the function
csv.file.name <- paste0("result_linkage_",id,".csv")
write.csv(one.year(id), csv.file.name, row.names = FALSE)

/// END OF FILE ///
```

aggregate.batch file:

```
#!/bin/bash
#SBATCH -p shared
#SBATCH --mem 8g
#SBATCH -t 0-6:00
#SBATCH -c 5
#SBATCH -o out_%A_%a.out  # standard output
#SBATCH -e err_%A_%a.out  # standard error

## EXECUTE CODE ##
singularity exec geospatial_latest.sif R CMD BATCH aggregate.R script_aggregate

/// END OF FILE ///
```

aggregate.R file:

```
#file for aggregating yearly data

#all years considered
years <- seq(2000,2018)

#create list to store results for each year
aggregated.results <- vector(mode="list", length = length(years))

#initialize loop counter
i <- 1

#for loop to create yearly data frames
for (y in years){

    #generate the file name to open
    csv.file.name <- paste0("result_linkage_",y,".csv")

    #read the csv file and assign it to the master list
```

```
    aggregated.results[[i]] <- read.csv(csv.file.name)

    #advance counter
    i <- i + 1
}

#condense the results into 1 data frame which also auto sorts it by State
condensed.results <- Reduce(function(x,y) merge(x = x, y = y, by = "State"), aggreg
ated.results)

#create a csv file from the condensed data
write.csv(condensed.results, file = "result_linkage.csv", row.names = FALSE)

/// END OF FILE ///
```

Step 2: Next I copied the 4 files to my home directory on Cannon:

```
scp ./problem_set/problem4rcode.R e88bu2044@login.rc.fas.harvard.edu:~/problem4
scp ./problem_set/problem4.batch e88bu2044@login.rc.fas.harvard.edu:~/problem4
scp ./problem_set/aggregate.R e88bu2044@login.rc.fas.harvard.edu:~/problem4
scp ./problem_set/aggregate.batch e88bu2044@login.rc.fas.harvard.edu:~/problem4
```

Step 3: Open a compute node:

```
Access 1 compute node with 4g memory, for 6 hours on the class's shared partition:
[e88bu2044@boslogin02 problem4]$ srun -p shared -n 1 -t 00-6:00 --pty --mem=4000 ba
sh
```

Step 4: Create a batch job array

```
[e88bu2044@holy7c18412 problem4]$ sbatch problem4.batch
This creates .csv files for every year by splitting each year into its own job in t
he array
Error files were all empty and my directory contained all the correct files after t
he job array was completed
```

Step 5: Aggregate the data

```
[e88bu2044@holy7c18412 problem4]$ sbatch aggregate.batch
This reads in all the yearly csv files and aggregates them into one csv file
```

Question 4.4: Download data from Harvard Cannon

Copy `result_linkage.csv` to your `data/` folder in Renku.

Question (4.4): You have generated `result_linkage.csv` file on Harvard Cannon in Question 4.3. Now, the next and last step is to copy this file to your `data/` folder in your Renku `problem3` folder. Describe the steps and commands that we are using and make sure your run the code snippet below.

Your answer (4.4):

I entered the following from a Renku terminal (while a SSH connection was open):

```
$ scp e88bu2044@login.rc.fas.harvard.edu:~/problem4/result_linkage.csv ./data

Breakdown of the command elements:
    scp: secure copy command
    e88bu2044@login.rc.fas.harvard.edu: username and server where the file is locat
ed
    ~/problem4/result_linkage.csv: location of the file on the server
    ./data: location where I am copying to
```

Make sure you run the following code snippet:

In [1]:
```
d <- read.csv("../data/result_linkage.csv")
d
```

A data.frame: 52 × 20

State	PM25_2000	PM25_2001	PM25_2002	PM25_2003	PM25_2004	PM25_2005	PM25_2
<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<d
Alabama	15.347304	12.882067	12.184880	12.965523	12.613791	13.510309	12.810
Alaska	1.163441	1.436871	1.503111	1.561109	3.068461	2.123017	1.800
Arizona	5.689456	5.993380	5.117804	5.184043	3.938154	4.312784	4.530
Arkansas	12.217614	10.674957	9.675823	10.481907	9.085027	11.124031	9.643
California	7.235699	7.365023	7.532225	6.398249	6.003756	5.925828	6.118
Colorado	5.130564	5.214317	5.001115	4.398830	4.030641	3.966444	3.918
Connecticut	10.639409	11.514075	10.585961	10.430752	9.432230	9.420129	8.167
Delaware	14.472174	13.955687	13.521107	13.271980	12.842338	13.533627	12.388
District of Columbia	16.108197	15.863388	15.031694	14.154098	14.377596	14.980328	13.393
Florida	11.992685	10.026597	9.427851	9.660862	10.583474	10.565889	10.101
Georgia	15.298356	12.611452	11.931417	11.892146	12.733371	12.953578	12.571
Hawaii	NA	NA	NA	NA	NA	NA	
Idaho	5.310285	3.937207	3.868958	3.854266	3.878750	4.185322	4.255
Illinois	11.669739	12.038823	10.994932	11.000167	9.621084	12.672150	10.004
Indiana	13.340957	13.176687	12.819609	12.756080	11.025661	14.264042	11.674
Iowa	8.699789	8.915940	8.712073	9.830617	8.358763	10.203216	8.481
Kansas	6.979502	6.981876	6.685182	6.728385	6.063742	7.044378	6.277
Kentucky	13.590258	13.125039	11.725765	11.916318	11.197411	13.600271	11.870
Louisiana	12.069336	10.422241	9.439268	10.746674	9.771054	11.827060	10.845
Maine	4.344670	4.891472	5.057838	5.760975	4.751804	5.255351	4.397
Maryland	13.698000	13.355853	12.745050	12.331229	12.490721	13.156707	11.659
Massachusetts	8.858266	9.410097	8.793990	8.859371	8.287843	8.060651	7.182
Michigan	7.036112	7.677911	7.882109	8.150643	7.275905	8.870612	7.403
Minnesota	6.211781	6.086538	5.800740	6.782576	6.665649	6.752646	6.007
Mississippi	12.518602	10.927095	10.406771	11.130582	10.633436	11.996163	11.216
Missouri	9.645852	9.473839	9.400547	9.595661	8.090892	10.505908	8.613
Montana	4.517315	3.570535	3.165464	3.985604	2.941988	3.128447	3.648
Nebraska	6.018922	5.496944	5.299472	5.434019	4.884213	5.381013	4.850
Nevada	4.592638	4.946056	3.854972	3.802235	3.694125	4.073471	3.778
New Hampshire	6.030525	6.807319	6.760655	7.202446	6.300391	6.344159	5.541
New Jersey	13.977770	13.596591	12.438971	12.856051	11.750664	12.751560	11.009
New Mexico	4.837870	5.011119	4.456524	4.376730	3.465059	3.866692	3.670

State	PM25_2000	PM25_2001	PM25_2002	PM25_2003	PM25_2004	PM25_2005	PM25_2
<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<d
New York	8.584933	8.905972	8.817709	9.081832	7.805046	8.767291	7.183
North Carolina	13.190457	12.060941	11.479315	11.107806	11.936535	12.296690	12.203
North Dakota	5.139933	4.318172	3.931727	4.476046	4.118152	4.485981	4.419
Ohio	13.254557	12.922059	12.938014	12.900991	11.815539	13.443589	10.991
Oklahoma	8.027203	7.948275	7.336305	8.131242	7.582253	8.952467	7.327
Oregon	3.302825	3.180348	3.580155	3.221450	3.530344	3.485656	3.645
Pennsylvania	11.823097	12.001746	11.275934	11.330120	10.936986	11.423820	9.636
Puerto Rico	NA	NA	NA	NA	NA	NA	
Rhode Island	9.621045	10.530667	9.225708	9.471576	8.805519	8.730579	7.820
South Carolina	14.522920	12.323125	11.480108	11.386770	12.230113	13.174238	12.923
South Dakota	5.392872	5.184528	4.469669	4.791280	4.551043	5.008997	4.847
Tennessee	13.883759	12.363131	11.579901	11.684802	11.628637	12.944490	12.082
Texas	8.570381	7.902247	7.335994	7.935208	6.907435	8.016654	7.236
Utah	4.961843	5.826140	4.519672	4.394711	4.645198	4.531372	4.085
Vermont	5.793251	6.335894	7.240539	7.491725	6.608264	6.944198	5.665
Virginia	12.539216	11.834776	10.686814	10.565359	11.013120	11.652073	10.593
Washington	4.651522	4.364985	4.354338	4.591031	4.681852	4.043883	4.437
West Virginia	12.254838	12.231564	10.469770	10.716321	11.241424	12.200484	10.158
Wisconsin	7.624934	8.054028	8.071813	8.647736	7.496344	8.397782	7.117
Wyoming	4.569579	4.653051	3.833058	3.967193	3.980457	4.154461	4.114

Self-assessment (not graded)

Questions

1. What do you think I was hoping for you to learn through this homework?
2. Did you find anything particularly challenging?

Your answer

1. How to use a cluster, rocker, scripts, run job arrays, and how to aggregate "big data".
2. Aggregating the files was very challenging. I had to totally rethink the way I generally program loops and get outputs from functions.

Submitting your work

To submit the problem set, export the notebook to HTML and upload the file to Canvas.

[File -> Export Notebook As... -> Export Notebook to HTML]