Instructions for DCC access:

* Open terminal (on a mac, hit command + space then type “terminal”)
* Enter ssh NETID@dcc-slogin.oit.duke.edu
  + NETID is your net id, e.g. gt83 for me.
* Follow instructions for entering your password and doing 2 factor authentication. If you login to the VPN (Cisco AnyConnect) then you won’t need to do two factor authentication.
* Check out where you are by typing: ls
  + This will list all the files in the folder you are currently in
* Navigate to our class directory: cd /hpc/group/sta440-f20
* Check contents with: ls
  + Should be a VOTER and WESAD folder (plus gt83, that’s me!, and anyone else who has made their directory so far)
* Create a folder: mkdir NETID
  + Again, NETID is your netid.
* Go into that folder: cd NETID
* Check folder is empty: ls
  + You should see NOTHING.
* WITHOUT GITHUB:
  + Make an R file: touch checkout \_data.R
  + Open it for editing: emacs checkout \_data.R
  + Type the following commands:
    - print(“File loading…”)
    - data1 <- read.csv(“../lab\_data/person1.csv”)
    - print(head(data1))
  + To exit and save: Control + xc
* WITH GITHUB
  + To Do
* Start an interactive session: srun --pty bash -i
* (the above may take a few moments to execute if the cluster is very busy)
* Load R: module load R/3.6.3 (if you don’t specify the version, the most recent version will be loaded, 4.0.0)
* Start an interactive R session: R
* You can now run any R commands you like! Installing packages from here is generally how I do it.
* Check that the script is working by typing: source(“checkout\_data.R”)
* To quit R: q()
* To exit the interactive session: exit
* You now want to make an R script that does exactly what we just did in our interactive session.
* Make the file: touch execute.R
* Edit the file: emacs execute.R
* Type the following:
  + print(“starting execution”)
  + source(“checkout\_data.R”)
  + print(“finished execution”)
* Hit control + xc to quit and save the file.
* It may seem silly to run this execute.R script which just runs a single other script, but this will be much more helpful when the project gets more complicated and you have separate scripts for data cleaning, data processing, and analysis.
* Now we need to make a shell script that will tell the server how to run execute.R
* Make the file: touch submit\_job.sh
* Edit the file: emacs submit\_job.sh
* Type the following lines
  + #!/bin/bash
  + #SBATCH --account=sta440-f20
  + #SBATCH -p common
  + #SBATCH -N1
  + #SBATCH -c1
  + #SBATCH --mem=1G
  + module load R/4.0.0
  + Rscript execute.R
* Quit and save emacs with control + xc
* Now you are ready to submit a job! Type the following command: sbatch submit\_job.sh
* You’ll see a notification along the lines of: Submitted batch job 43496862. That number is the job number.
* To see if your jobs have been allocated resources type: squeue -u NETID
  + Where NETID is your netid. This will show you what jobs you have going currently.
* When your job finishes (it should be pretty fast) type: ls
  + You should see a file called slurm-NUMBER.out where NUMBER is the job number from above.
  + To see the file contents type: cat slurm-NUMBER.out
    - You can press tab for autocomplete of file names. Typing “slurm-“ then hitting tab should complete the file name.
  + The contents should be the print messages from all the code files and the dataset person1.csv

Now that you have walked through this example, you’re ready to write your own code! Your assignment is to execute on the server an R script that combines the data files person1.csv, person2.csv, and person3.csv into a dataset with 9 rows of the three variables from each .csv AND an additional 4th variable that is a person-id variable.