Homework 13. Due by 11:59pm on Sunday 12/3.

Statistical computing on greatlakes

From homework 11, we understand the relevance of being able to implement embarrassingly parallel calculations. In the context of statistical computing in R, we are motivated to learn to use `foreach` on the greatlakes cluster. In this homework we carry out a simple example, since many of us are new to Linux clusters. Once you can run a simple multicore `foreach` on greatlakes you are prepared for a large fraction of all statistical parallel computing tasks. Work through the questions below and write brief answers to the following questions, by editing the tex file available at `https://github.com/ionides/810f23`, and submit the resulting pdf file via Canvas.

At various points in this course we have seen the research value of working with command line tools and plain text file formats. These facilitate reproducibility and avoid additional user interface software. An introduction to command-line access to greatlakes is at `https://arc.umich.edu/greatlakes/user-guide/#document-3`.

User interface software (including integrated development environments for programming, such as RStudio) can boost productivity. However, a statistical researcher should be able to dig into the details of the computations they are carrying out. These are the very details that user interfaces aim to protect us from. A web interface to greatlakes can provide access to RStudio, Jupyter, and a remote Linux desktop (`https://arc.umich.edu/open-ondemand/`). Try this interface if you like, in addition to the command-line method.

1. R has a reputation of being convenient for data analysis but slow in some situations, both features shared with Python. Loops in R can be slow, though careful coding may avoid this (`https://adv-r.hadley.nz/control-flow.html#common-pitfalls`). If R run time starts limiting your research—i.e., whenever there is a numerical result you would like to see but you don't have the patience to wait for it to be computed—you have various options, including:

   (a) Use smaller examples.

   (b) Write the critical part of your code in C called from R. This can make use of the C functions underpinning the R language (`https://cran.r-project.org/doc/manuals/R-exts.html#The-R-API`). Alternatively, the Rcpp package facilitates combining R with compiled C++ code.

   (c) Work on rewriting your R code to make it more efficient.

   (d) Run your code on a more powerful machine, maybe moving from your laptop to greatlakes.

   Comment on situations when each approach might be suitable. Do you have advice, or relevant experiences?

YOUR ANSWER HERE.

2. Follow the instructions at `https://ionides.github.io/810f23/gl/slides.pdf` to run a job testing foreach on greatlakes. Report back on your results, and any problems that arose or advice to share.

   YOUR ANSWER HERE.