1 Description of Module 1

(From my own experience): Suppose you are part of a tech company's server analytics team. The team collects, among other things, data about storage capacity of servers, specifically the amount hard drive storage used per second (or minute). A figure below shows how the data would look like. I highlight some features of this data.

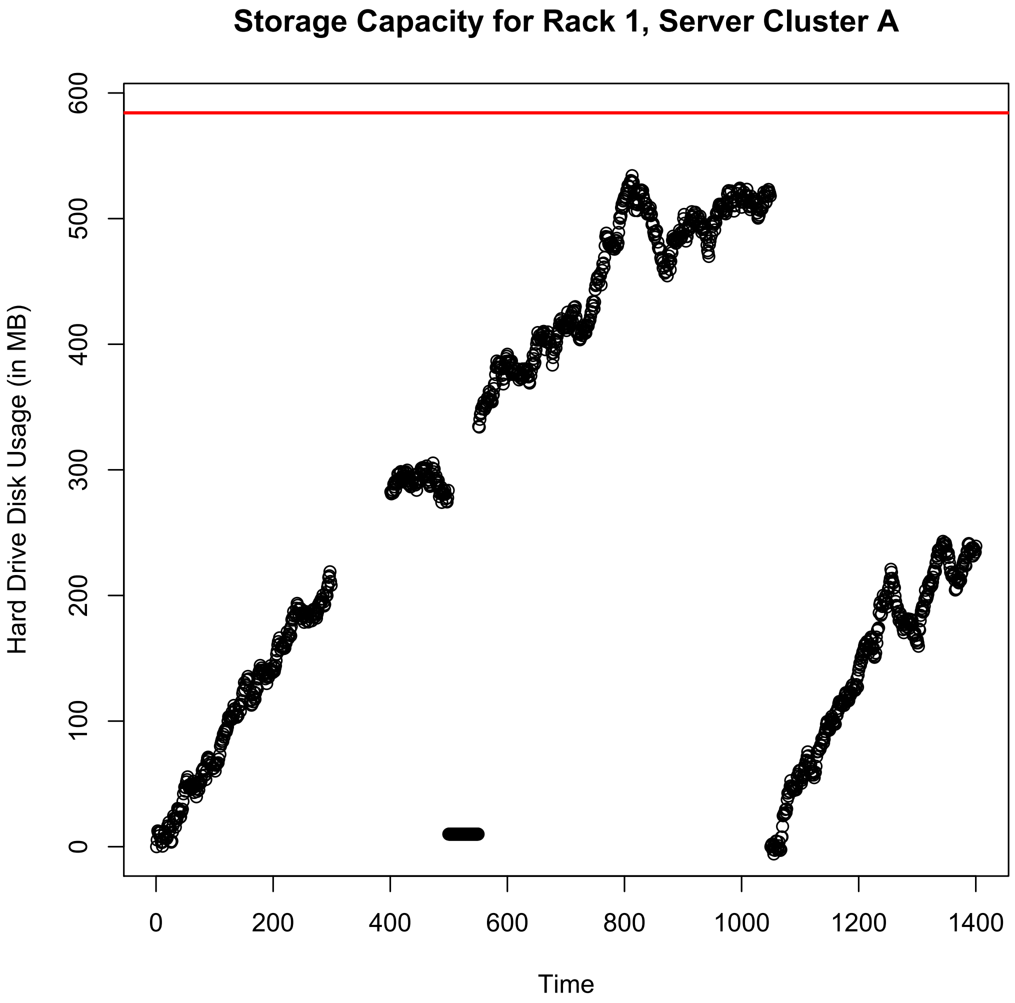

Figure 1: Status of Server Rack 1 in Cluster A. Each dot represents the amount of hard drive storage used at time t. The red line indicates the maximum storage limit for the hard drive.

1. Like your typical hard drive in your personal computer, each hard drive used in a server rack has a maximum capacity.

2. Although the trend is generally upwards, there is some stochastic behavior.

3. Around $t = 300$, the server was "down" and no data was collected.

4. Around $t = 500$, the server received software/hardware updates from an engineer and as a

result, the amount of storage used looks wonky during those times. These maintenance times are typically unplanned.

5. Around $t = 1100$, some engineer (or automated software) cleaned the hard drive so that it doesn't exceed the theoretical maximum capacity line.

You are tasked with **developing an algorithm that can predict when hard drives will reach its maximum capacity** so that engineers can plan ahead and make adjustments to the server storage. Some things to remember:

1. Server storage and CPU time are limited when running your proposed algorithm. You need to use the minimum possible software/packages/storage to run your code and use the minimum possible CPU time. For example, running lm() in R is very costly, both in terms of CPU time and storage. Running a faster version of lm() from RcppEigen package in R is also expensive because it requires installing (and eventually maintaining) an additional package. Also, the latter may interact poorly with the pre-existing server software.

2. Your algorithm needs to scale, both in terms of being able to handle large number of servers as well as being able to easily make updates to your algorithm in the future, especially after you leave the team.

3. Your algorithm must be highly robust to things like server failure, server down time, server maintenance time, etc.

4. Your algorithm needs to make fast and accurate predictions. In particular, server-side hardware engineers need at least a two-weeks notice about when storage would max out. Your algorithm should be able to tell the engineering team when a server rack would reach its maximum capacity before this time.

   It is never a good idea to go over the hard drive limit (or even approach this limit closely).

5. Your algorithm should be simple enough that a non-statistician (but an engineer) can understand how it is making the predictions.

2 Deliverables

You will work in the group that you've been assigned on Canvas. Your group will submit (1) a 2-page pdf summary that provides a concise, clear description of your algorithm and (2) an R code with the specifications described in the R file. In particular, your summary must provide (i) a working demo of your code (with realistic simulated data), (ii) some benchmarks for your code, and (iii) a concise, clear explanation of your prediction model. For the R code, we will run your R code against our checks that are described in the R file provided with the assignment.

Both (1) and (2) are due by **September 18, 2020 at 11:59pm CST**. You will submit both to Canvas. You are responsible for making sure that the two deliverables are submitted on time.