Statistically Random Subsets

Jarred Parr

August 2018

1 Introduction

This project laid out a fictional scenario of the acquisition of medical records in a data set that may or may not be randomized. The goal was to get k randomly selected records from a data set of size n and then return the sorted list. For this project the approach I took was the most efficient given the artificial constraints. For instance, instead of modifying the provided list, n, I instead had to copy the records over to a new vector since doing any of the operations in place would have lead to lost records. To avoid this, I simply created a new list to copy the elements over as to not lose any from the original list. As a result, the implementation lacks some minor core efficiency gains but it keeps with the project description.

2 What I Learned

Here I didn't learn a ton of new things about C++ itself since conceptually things were quite straightforward, however, I did learn a great deal about sorting algorithms and some high performance tuning in C++. On top of that, I implemented a quicksort algorithm from scratch for fun and to learn the inner workings of it. One could argue the built in sorting systems are better and I would not dispute that, but it was fun to learn how it worked on a deeper level.

3 Code Performance

My code exhibited the following run time performance:

- With the built in $std::sort$ function: $85ms$
- With my custom quick sort implementation: $115ms$

4 Conclusion

This project showed me a great deal about the basics of run time performance and how to do performance tuning to squeeze every ounce of speed out of a particular set of functions I have written. I really enjoyed learning what I did on this lighter project and am eager to tackle new ones.