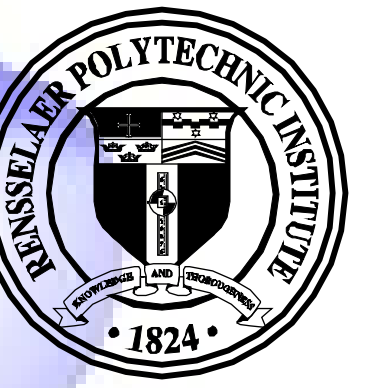




Closed Loop Optimization of Gene Sequencing Run Times



Daniel Casner^(1,3,4)

Jason Baumohl^(1,2), Alex Copeland^(1,2), Joshua Dillon^(1,3,5), David Pletcher^(1,3)

[1] DOE Joint Genome Institute, [2] Lawrence Berkeley National Laboratory, [3] Lawrence Livermore National Laboratory, [4] Rensselaer Polytechnic Institute, [5] Purdue University

On a high volume genome sequencing production line, optimizing throughput and removing bottlenecks are serious concerns. The goal of this project was to define a practical method for calculating the optimal duration of each sequencing run from historical data. Several methods for estimating the optimal run length of individual past sequences are presented as well as an algorithm for calculating the optimal run length for the population of plates. The algorithm accurately determines a shortened run time for low performing projects but also indicates that many projects should be run longer than they currently are.

Introduction:

The Joint Genome Institute (JGI) sequences more than 3 billion bases per year. One of the major bottlenecks in the sequence production process is the gene sequencing itself. Capillary electrophoresis is used so running the sequencer longer increases the number of bases read. However, later bases have lower quality scores and so may not be worth collecting.

In order to increase production at the JGI we want to find the optimal run time to collect all useful data without wasting time reading bases that aren't usable.

Previously we had no automated way to determine the appropriate run length. In this project we developed an algorithm for estimating optimal run length suitable for automation.



Figure 1: ABI 3730 Capillary Electrophoresis Gene Sequencer

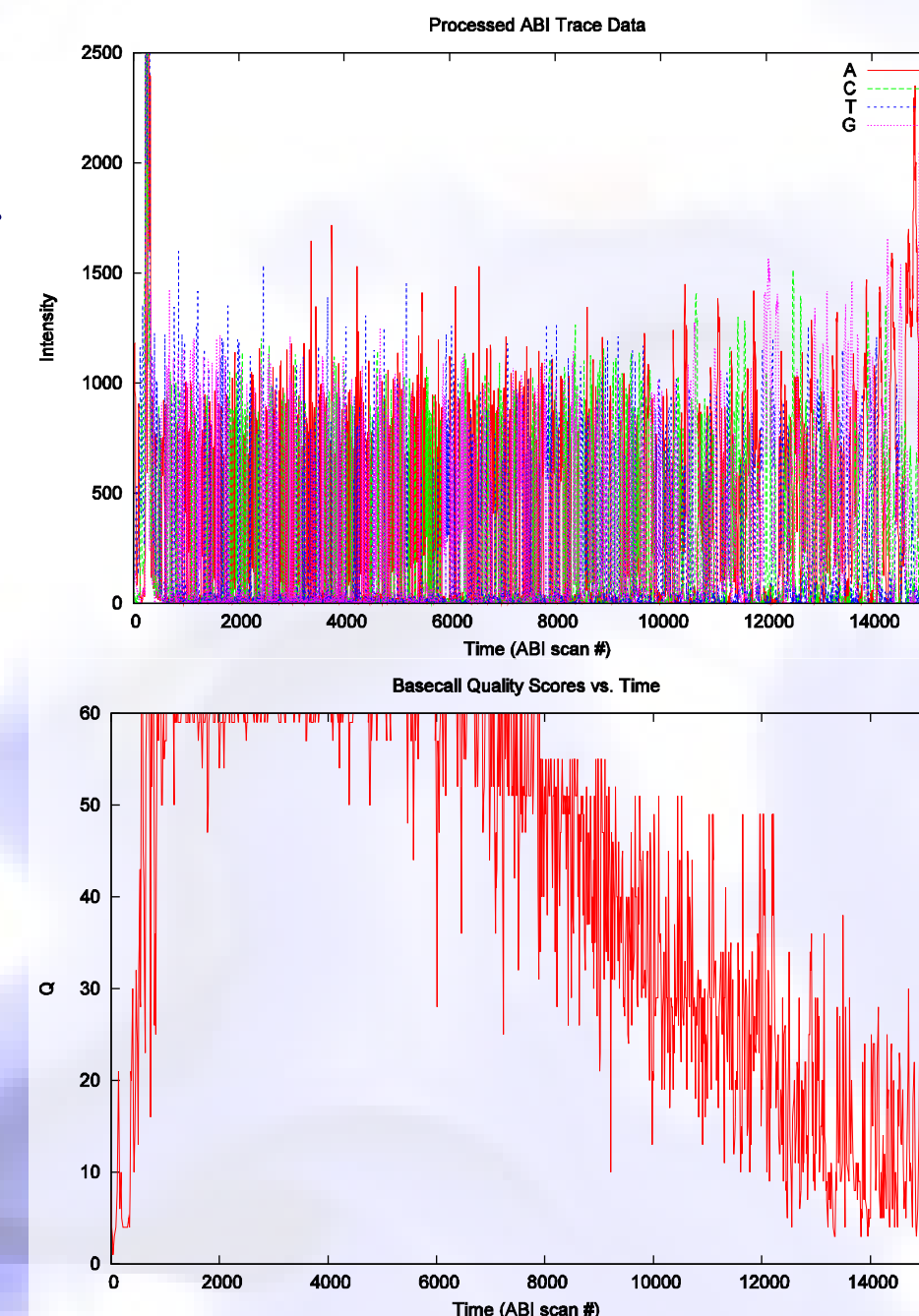


Figure 2a: Top: The processed trace data for all four bases from the ABI sequencer in a typical run. Bottom: The quality score of the base calls over the run. Quality is highest in the middle and falls off later in the run.

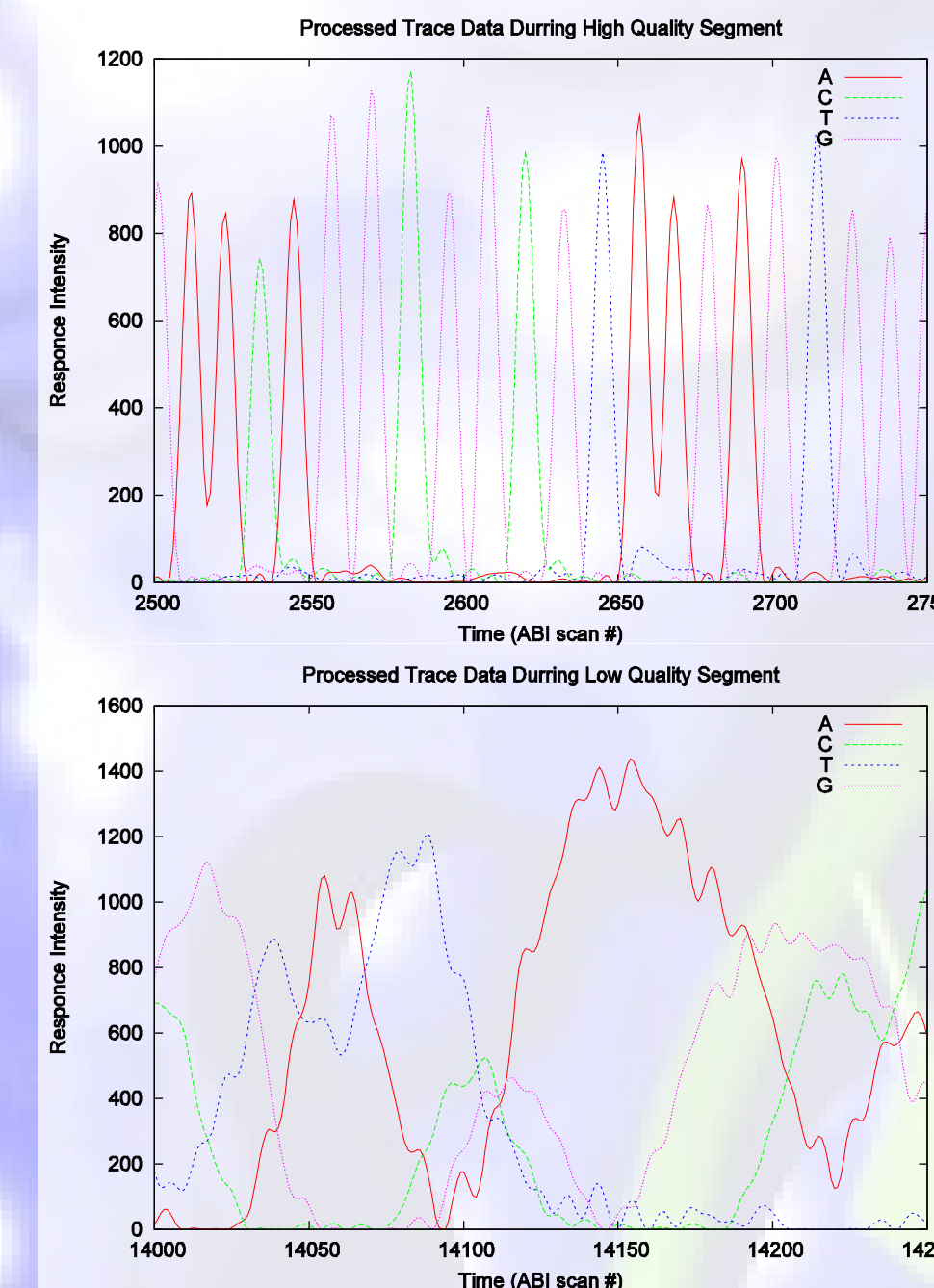


Figure 2b: An expanded view of the trace data during high and low quality parts of a run. Later in the run the signal gets noisier and bases aren't called with as great a confidence.

Methods:

Prior to looking for an algorithm to find the optimal stop time, we examined several metrics of the “end of useful data” including:

- Last call time of the KB base-calling algorithm
- Number of “Q20” bases
- Jazz trim length
- Trends in Q score of base-calls
- PHRAP alignment length

These metrics indicate the time when a sample which has already been sequenced, stopped producing useful data.

Once we have a collection of stop times for past samples, this data is clustered using the *k-means*, *farthest neighbors* and *nearest neighbors* algorithms. A convolution of Gaussian distributions is constructed based on the means and variances of the clusters. The convolution which fits the data best is then used as a continuous probability density function.

From the probability density function we can trivially calculate the necessary runtime to collect N-percent of the data by Riemann sum integration.

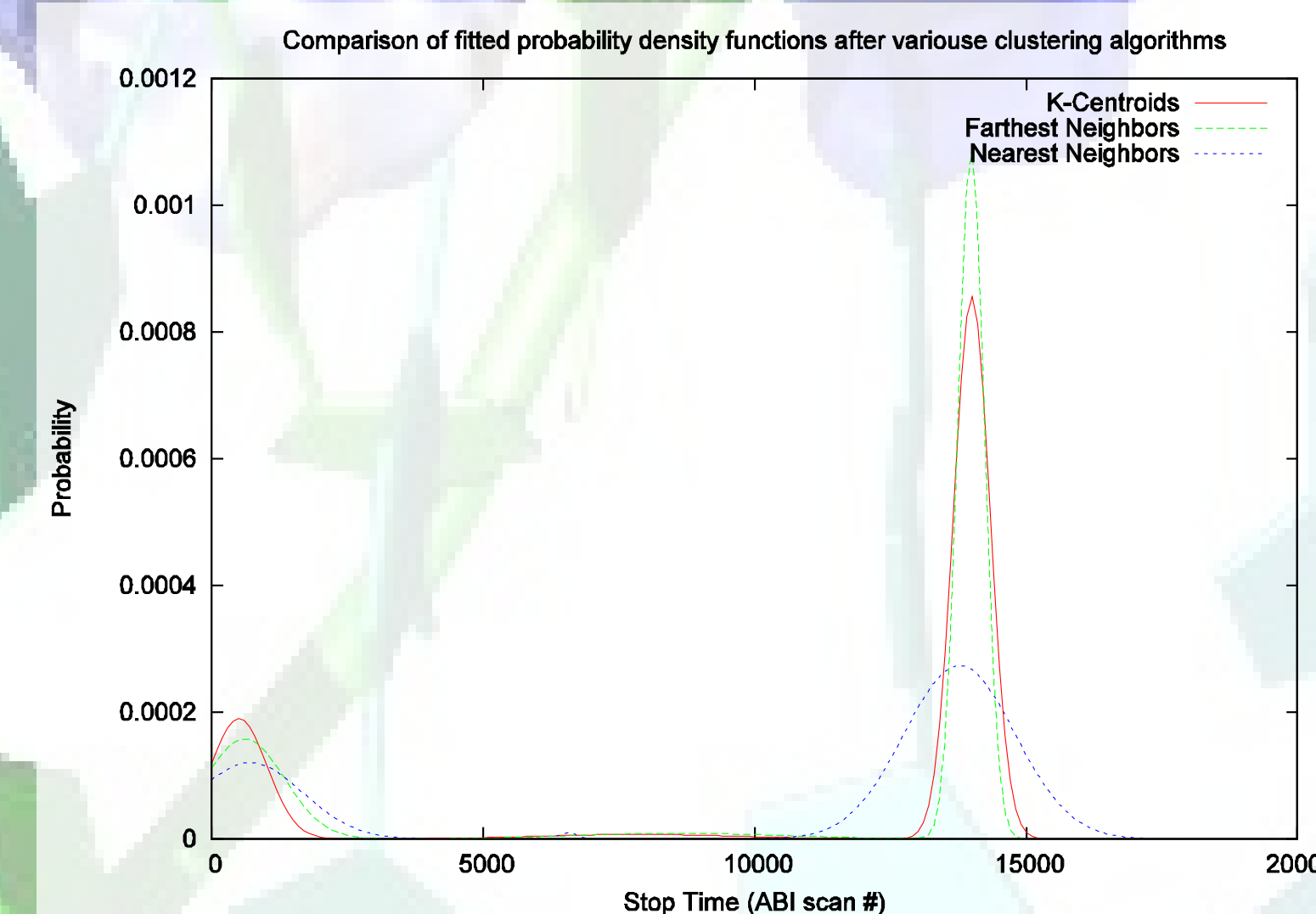


Figure 3: This plot shows the convolved Gaussian distributions resulting from all three clustering algorithms run with “end of useful data” times produced by a variant of the Jazz trimming algorithm.

The three clustering algorithms produce somewhat different results even when run on the same data. Hence we use all three and choose the one which best fits the data we have. Usually *Farthest Neighbors* provides the best fit but not always.

Results:

Two distinct cases emerged after running our algorithm on a number of libraries.

3. Low performing: The majority of the samples cease producing useful data before the end of the run time (e.g. Fig. 4a). In this case the algorithm fits sample data with low deviation and predicts an optimal stop time for the population which matches observed results.

5. High Performing: A large number of samples continue to produce data up to the end of the run and are only stopped when the sequencer is stopped (e.g. Fig 4b). The stop times form a large cluster at the end of the run time. Our algorithm fits a distribution from the variance of the samples which stop before the end of the run time. From this approximation a lengthened runtime is estimated.

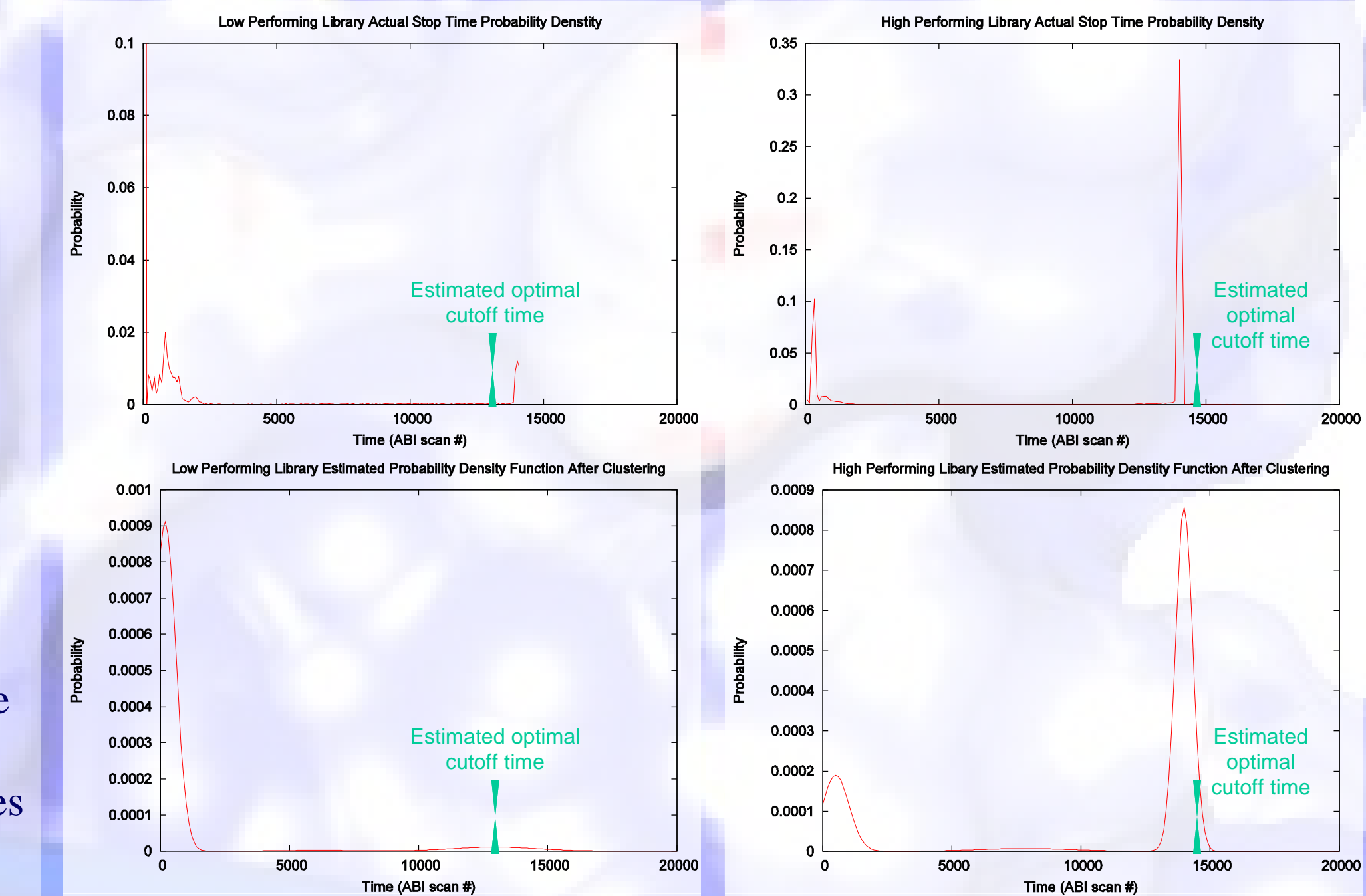


Figure 4a: A low performing library whose run time should be shortened.

Figure 4b: A very high performing library which would produce more data given a longer run time.

Original Run Length	14000 scans	78 minutes
Initial Estimated Optimal Run Length	14600 scans	81 minutes
Experimental Run Length	18000 scans	100 minutes
Estimated optimal run length after experiment	16600 scans	92 minutes

Table 1: Summary of our extended run time experiment.

In the second case finding the “optimal” run time is difficult because the algorithm has to extrapolate what the distribution would have looked like had the sequencer been run longer. To determine the validity of our algorithm in this case, we conducted an experiment where several plates were run longer than normal.

The estimated lengthened run time is not as long as the true optimal run time, however, it is sufficient to walk the run time out to optimality over several iterations. Since the cost of running the sequencers too long is far greater than the cost of discarding a small amount of data, this is a useful behavior.

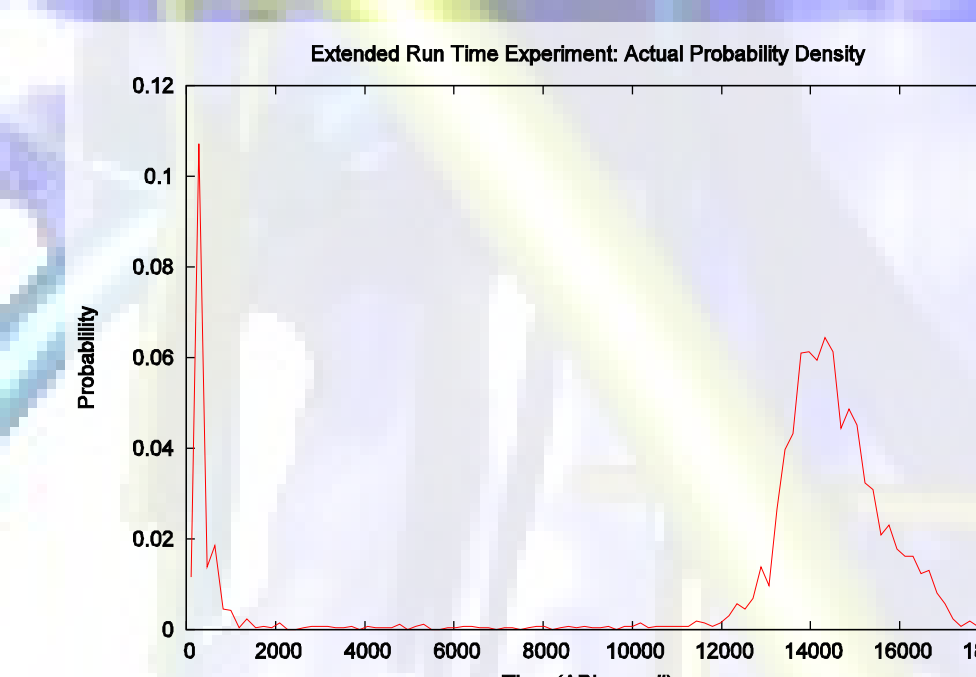


Figure 5a: Stop time data from the extended run time experiment.

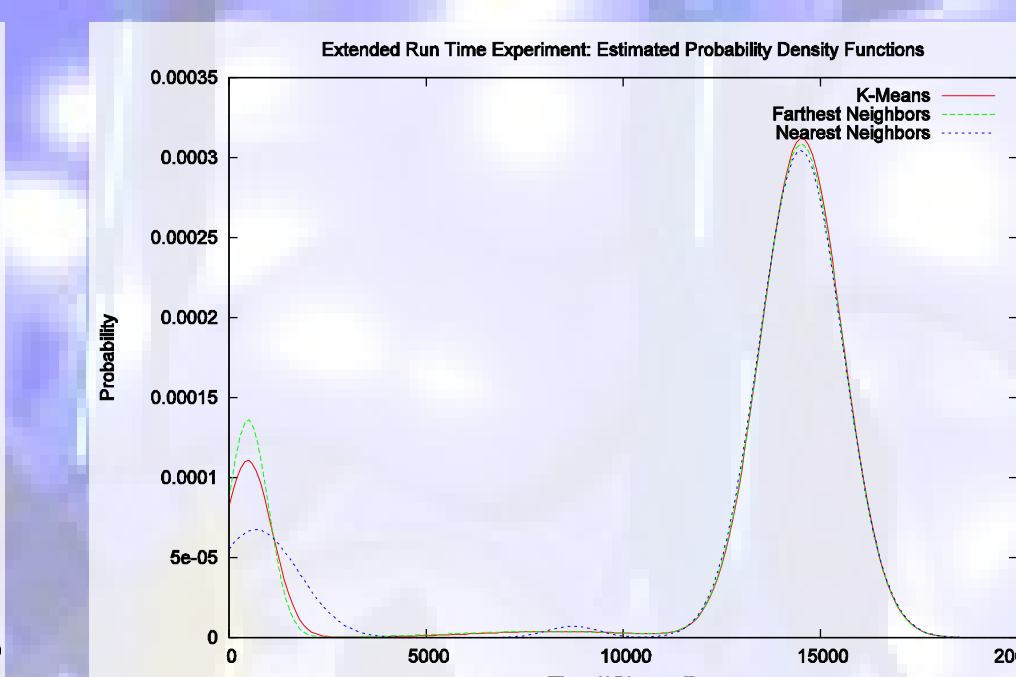


Figure 5b: The estimated probability density function for the experimental data. Note the difference between this and figure 4b.

Discussion:

Our original goal was to improve production by shortening sequencing run time where resources were being wasted collecting unusable data.

Ironically we find that we should run a large number of high performing libraries longer than we presently do. In order to find truly optimal run times, we need a cost model that can directly compare the cost of lost material and opportunity associated with longer run times to the cost of bases missed with shorter run times.

For low performing samples our algorithm does a very good job finding the optimal run time for the population.