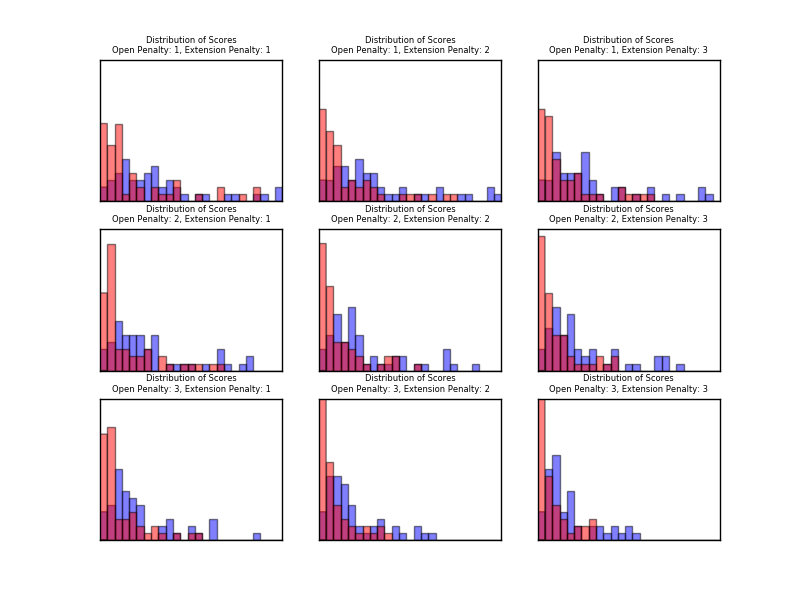
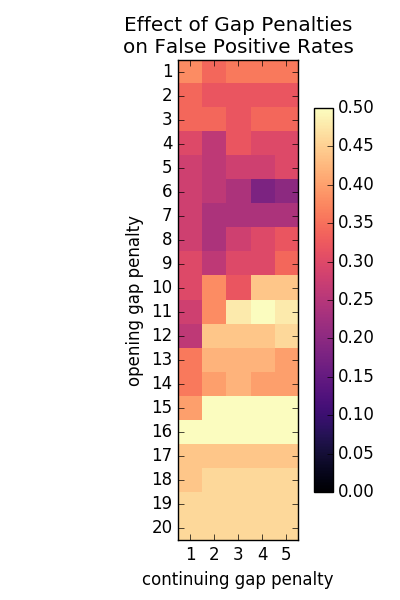
**Part 1A**

*Consider the false positive rate (proportion of negative pairs with scores that exceed a score threshold) when the true positive rate (proportion of positive pairs with scores above the threshold) is 0.7. What's the best false positive rate that you can achieve with varying both gap opening (from 1 to 20) and extension penalties (from 1 to 5) with the BLOSUM50 matrix? What is the best gap penalty combination?*

For a quick preliminary analysis, I generated histograms of alignment scores for our positive and negative datasets (left). The general goal of this part of the project is finding a scoring strategy that maximizes the separation of the two histograms.

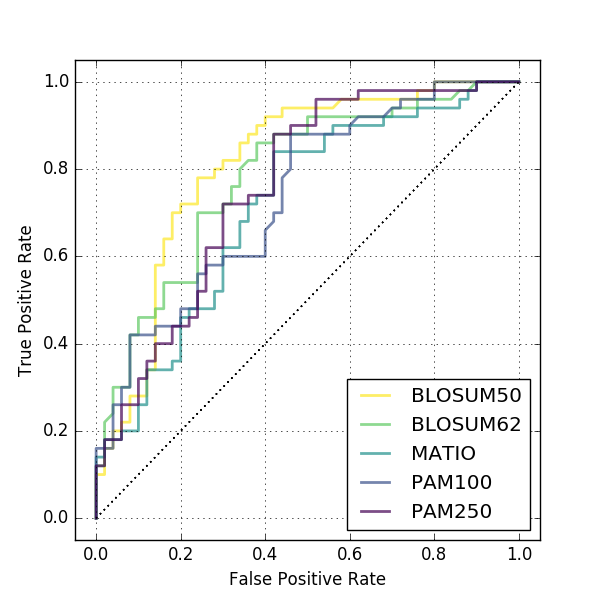
Specifically, we can set a threshold at the score at the point above which 70% of the positive scores are found. Quantifying the number of negative scores that exceed this threshold gives a general measure of the quality of separation.

For alignment of these particular sequences with the BLOSUM50 matrix, this false positive rate is shown above (right). The optimal penalties are 6 and 4 for the opening and extension penalties, respectively. A large opening gap penalty seems to be detrimental to the alignment quality in this particular case.

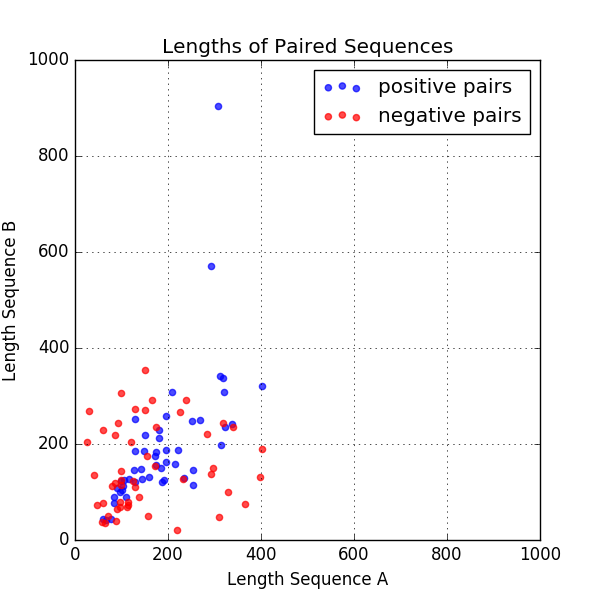
**Part 1B**

*Using the gap penalties you determined from question 1, which of the provided scoring matrices performs the best, in terms of false positive rate (at a true positive rate of 0.7)? What are the performance rates of each of the matrices? Create a Receiver Operator Curve (ROC) graph which shows the fraction of true positives on the Y axis and the fraction of false positives on the X axis. Include on the graph data for each of the provided matrices.*

Given the BLOSUM50-optimized gap penalties, it is perhaps unsurprising that the BLOSUM50 matrix performs best out of the given scoring matrices. If the optimal gap penalties were used for each matrix, this may not have been the case.



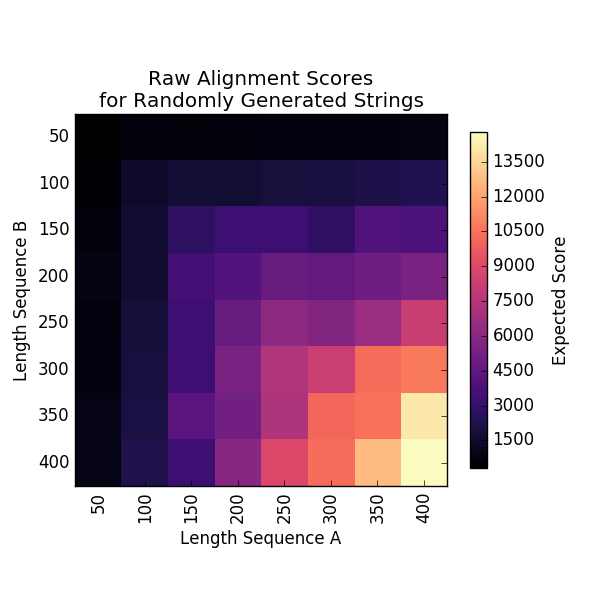
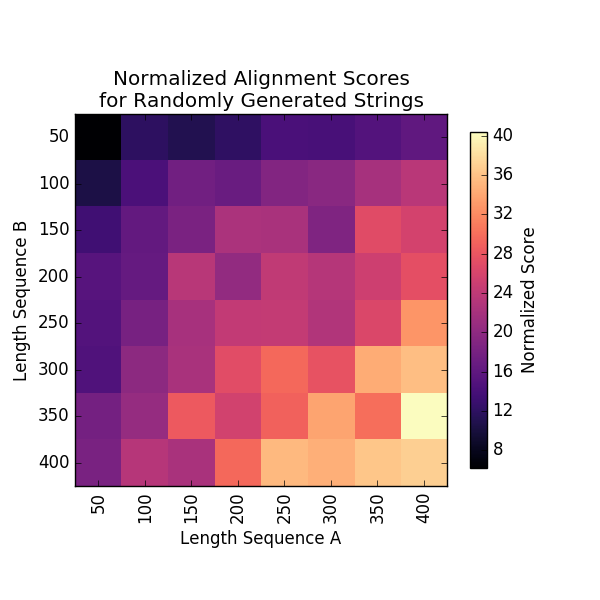
**Part 1C**

How does the performance change if you normalize the Smith-Waterman scores by the length of the shortest sequence in a pair (i.e. divide the raw score by the min length)? Show the ROC curves for your best matrix and for the same matrix with normalized scores. Are the false positive rates better or worse? Why do you think this is so?

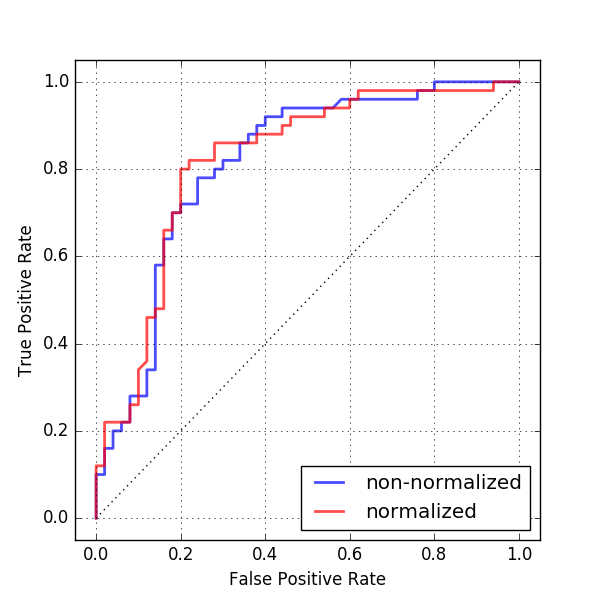
The dependence of the alignment score on the length of input sequences may be relevant to analysis of the sample sequences we’re using to optimize alignment patterns. For most pairs, the sequences both sequences have less than 400 amino acids. Interestingly, positive pairs tend to have more similar lengths than negative pairs.

In order to evaluate the impact of length I generated a null distribution of scores using randomly generated sequences of varying lengths.

This null distribution can be generated for a particular scoring system by scoring randomly generated amino acid sequences of particular lengths. For the previously optimized conditions, the average score of 25 randomly generated strings is shown below. The dependence of the score on the length of the input sequences is obvious in this figure. Longer random strings paired with other long random strings have inflated scores. To correct for this completely, we could normalize real scores by generating a corresponding expected value, but this is computationally expensive. It is simpler and potentially adequate to divide the raw scores by the minimum length of the input sequences. This somewhat corrects the length biases as shown below (i.e. it results in a flatter distribution across input lengths).

Unfortunately, this normalization does not appear to greatly improve the ROC of our optimized alignment system, though there is some improvement around true positive rates of 0.8.

This may be due to the strings being nonrandom, or due to the imperfect normalization method. Perhaps a

**Part 2A**

*Devise an optimization algorithm to modify the values in a starting score matrix such as to maximize the following objective function: sum of TP rates for FP rates of 0.0, 0.1, 0.2, and 0.3. The maximum value for the objective function is 4.0 (where you are getting perfect separation of positive and negative pairs even at the lowest false positive rate). You should use the gap and extension penalties derived from Part 1. Remember, you must maintain symmetry in your matrix. You can make use of real-valued scores in the matrices if desired (this is probably a good idea).*

I chose to use a genetic algorithm to optimize the scoring matrix. Each ‘epoch’, many slightly altered matrices are generated. At the end of an epoch, the best-scoring matrix is taken as the new starting point and the process begins again. The method quickly maximizes the objective function for the BLOSUM50 matrix and produces a significant improvement in the ROC curve for the static alignment.

As written, my optimization strategy only works on the static alignment. This is not realistic since changing the scoring matrix, will actually change the alignments. In fact, realigning the dataset using the static-alignment-optimized scoring matrix shows no improvement.

The same general strategy could be used to iteratively improve the resulting alignments of

**Part 2B**

*Beginning from the best matrix from above (that which produced the alignments), run your*

*optimization algorithm to maximize the fitness of the new matrix. How much improvement do*

*you see in the fitness? Show the full ROC curves for the original matrix and the optimized*

*matrix. What happens when you now realign the sequences using the new matrix and*

*rescore? Show the new ROC curve following realignment on the same graph as above.*

*Qualitatively discuss how your matrix and your alignments change following optimization.*

**Part 2C**

*Beginning from the MATIO matrix, but using the same initial sequence alignments, re-run*

*the optimization. Show the same ROC plots as for (2). Discuss the relationship between the*

*results you see here and the results you saw for (2).*

