1) For both the *motif-distribution* and *sequence-distribution* as the number of processors P increased, for a given set of input files, the speed-up seemed to follow the pattern of continuously increasing since as the number of processor increases, the more work we are getting done in parallel since each processor is spending less time comparing the individual characters.

2) For both the *motif-distribution* and *sequence-distribution* as the number of processors P increased, for a given set of input files, the efficiency seems to either increase and then decrease, or decrease. I would attribute this to the observation that as we increase the number of processors, although we may be able to get more work done, the overhead associated with "managing" the increased processor count also increases with impacts our efficiency in a negative manner.

3) Seeing as though the average efficiency for the *motif-distribution* is 0.668387 while the average efficiency for the *sequence-distribution* is 0.902441, I would have to say that the *motif-distribution* appears to be the more efficient approach out of the two.