Genetic Relationship of Endangered Species

New Mexico

Supercomputing Challenge

Final Report

April 4, 2007

Team #82

Oñate High School

Team Members:

Kevin Christeson

Natalie Salvat

Justin Atteberry

Reese Davies

Teacher/Mentor

Donald Downs

Josefina Dominguez

Table of Contents

- Executive Summary--P. 3
- Introduction--P. 4
- Description---P. 6
- Results---P. 10
- Conclusion--P. 15
- Program--P. 16
- Most Significant Original Achievement---P. 24
- Acknowledgements--P. 25
- Appendix--P. 26
- Bibliography---P. 27

Executive Summary

Does a genetic similarity exist in endangered New Mexican animals? To solve this problem, we created a program to globally align twenty-eight different protein sequences, fourteen endangered and fourteen control species, also native to New Mexico. Bioinformatics, a new field of computational biology, enables us to perform a global alignment on a large amount of sequences efficiently. To perform this alignment quickly and accurately, we used a proven method known as the Needleman-Wunsch algorithm. This scoring method enables us to perform the 784 necessary alignments with relative ease. To easily read our results, the final scores were placed in individual charts; endangered vs. endangered, endangered vs. non-endangered, and non-endangered vs. non-endangered. The result is difficult to determine from these matrices, so we took the average score from each region, and from these results it was determined that endangered animals do have a higher average genetic similarity than the non-endangered control set. The results obtained in our experiment were significant, but a more detailed statistical test is necessary to verify our findings. This test will be performed before the final presentations in Los Alamos, and will make our findings more meaningful.

Introduction

For years, more and more animals around the world have become extinct for reasons such as changes in weather, loss of habitat, human intervention, etc. However, is there a possibility that there is more to extinction than environmental changes or acts of man? Is it possible that all endangered animals share genetic similarities? Our problem involves creating a program that will compare the genetic code of endangered and non-endangered animals to determine if any genetic similarity exists. We have restricted the animals we are testing to those found in New Mexico. Furthermore, a control set of non-endangered animals will be used to verify our results.

In order to solve our problem one must understand the science of genetic similarities. We are looking for similarities in the protein sequences of each of our species. These sequences are formed by the process of polymerization. Polymerization is the chemical process that bonds the amino acids into a protein sequence that is unique to each organism. Addition polymerization, or chain growth, is the primary method amino acids undergo to form protein chains. We will compare each species' genetic sequence to all other species using a new branch of biology known as bioinformatics.

Bioinformatics is the computational branch of molecular biology. Bioinformatics has been the center of some of the most recent breakthroughs in biology, such as the completion of the human genome project, new biotechnologies, new legal and forensic techniques, and new medicine for the future. Amino acids, which are complex organic molecules, are the basic building blocks of protein sequences. Biochemists realized that these protein sequences were huge molecules (macromolecules) made up of large numbers of amino acids.   They came up with names and three letter codes for twenty of the amino acids such as Alanine or Ala, Arginine or Arg, and so forth. A scientist then discovered that any given protein sequence always contains precisely the same number of total amino acids in the same proportion.   Finally, scientists discovered that amino acids are linked together as a chain, and that the identity of a protein chain not only comes from the amino acids it is made up of, but also from the order of the amino acids.   Since the discovery of protein chains, analyzing them has been a central topic of bioinformatics.

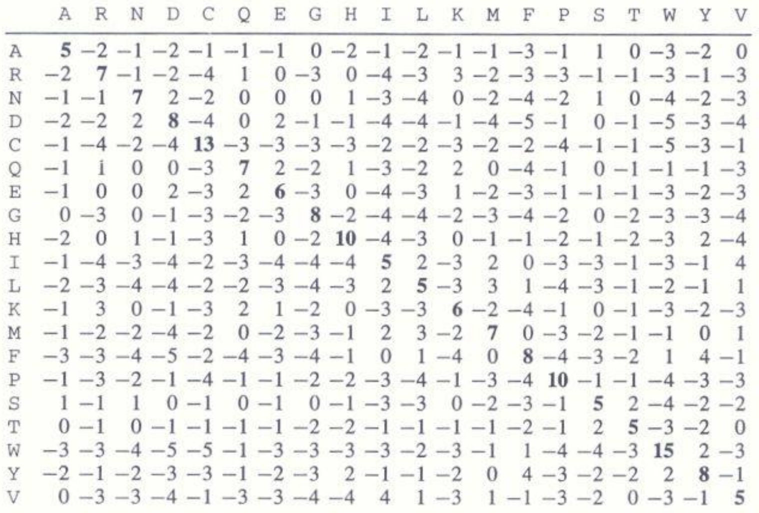
In our project, we plan to compare twenty-eight different New Mexico native animals to each other, on the molecular scale.   Fourteen of the animals will be non-endangered and the other fourteen will be endangered. We are not only comparing endangered to endangered to see if we can't find a genetic similarity, but also compare them to the non-endangered species for a control.   Utilizing the BLOSUM 50 Matrix, which compares each amino acid from two species and gives a result, we were able to build a program that compares all twenty-eight animals to each other. With the results from the program, we hoped to establish a connection between the endangered animals, which might point out that there is a genetic similarity between endangered animals.

Description

In order to align 28 different protein sequences of New Mexican wildlife with each other, one must use an efficient, proven algorithm. Bioinformatics contains a method known as global alignment, where the sequences are quickly and accurately scored. Global alignment involves constructing a two-dimensional matrix, and is built to allow “gaps”, or different sequence lengths to be acceptable. We construct a matrix ***F*** indexed by ***i*** and ***j***, where the value ***F(i , j)*** is the score for the two respective amino acids being compared. We find ***F(i , j)*** using the Needleman-Wunsch Algorithm:

$$F(i,j) = \begin{cases} F(i-1,j) + gappenalty \\ F(i-1,j-1) + s(i,j) \\ F(i,j-1) + gappenalty \end{cases}$$

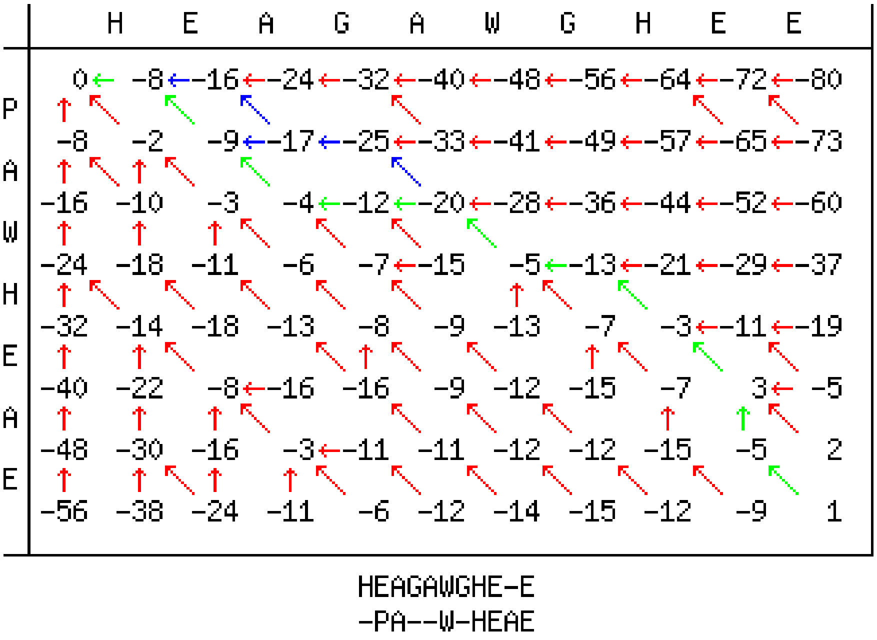
where ***s(i , j)*** is the score referenced in the BLOSUM50 scoring matrix, a commonly accepted tool used in computational biology:



	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	**5**	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−3	−2	0
R	−2	**7**	−1	−2	−4	1	0	−3	0	−4	−3	3	−2	−3	−3	−1	−1	−3	−1	−3
N	−1	−1	**7**	2	−2	0	0	0	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
D	−2	−2	2	**8**	−4	0	2	−1	−1	−4	−4	−1	−4	−5	−1	0	−1	−5	−3	−4
C	−1	−4	−2	−4	**13**	−3	−3	−3	−3	−2	−2	−3	−2	−2	−4	−1	−1	−5	−3	−1
Q	−1	1	0	0	−3	**7**	2	−2	1	−3	−2	2	0	−4	−1	0	−1	−1	−1	−3
E	−1	0	0	2	−3	2	**6**	−3	0	−4	−3	1	−2	−3	−1	−1	−1	−3	−2	−3
G	0	−3	0	−1	−3	−2	−3	**8**	−2	−4	−4	−2	−3	−4	−2	0	−2	−3	−3	−4
H	−2	0	1	−1	−3	1	0	−2	**10**	−4	−3	0	−1	−1	−2	−1	−2	−3	2	−4
I	−1	−4	−3	−4	−2	−3	−4	−4	−4	**5**	2	−3	2	0	−3	−3	−1	−3	−1	4
L	−2	−3	−4	−4	−2	−2	−3	−4	−3	2	**5**	−3	3	1	−4	−3	−1	−2	−1	1
K	−1	3	0	−1	−3	2	1	−2	0	−3	−3	**6**	−2	−4	−1	0	−1	−3	−2	−3
M	−1	−2	−2	−4	−2	0	−2	−3	−1	2	3	−2	**7**	0	−3	−2	−1	−1	0	1
F	−3	−3	−4	−5	−2	−4	−3	−4	−1	0	1	−4	0	**8**	−4	−3	−2	1	4	−1
P	−1	−3	−2	−1	−4	−1	−1	−2	−2	−3	−4	−1	−3	−4	**10**	−1	−1	−4	−3	−3
S	1	−1	1	0	−1	0	−1	0	−1	−3	−3	0	−2	−3	−1	**5**	2	−4	−2	−2
T	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	2	**5**	−3	−2	0
W	−3	−3	−4	−5	−5	−1	−3	−3	−3	−3	−2	−3	−1	1	−4	−4	−3	**15**	2	−3
Y	−2	−1	−2	−3	−3	−1	−2	−3	2	−1	−1	−2	0	4	−3	−2	−2	2	**8**	−1
V	0	−3	−3	−4	−1	−3	−3	−4	−4	4	1	−3	1	−1	−3	−2	0	−3	−1	**5**

Figure 2.2 *The* BLOSUM50 *substitution matrix. The log-odds values have been scaled and rounded to the nearest integer for purposes of computational efficiency. Entries on the main diagonal for identical residue pairs are highlighted in bold.*

The scoring matrix used also determines the gap penalty, and the BLOSUM50 requires a penalty of –8. This method builds recursively, starting with ***F(0 , 0)***=0. We also insert a gap in front of each sequence to allow the constant zero. Building from the top left to the bottom right of the matrix is difficult to perform manually, but entirely possible. One example might look like this:



		H	E	A	G	A	W	G	H	E	E
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
P	-8	-2	-9	-17	-25	-33	-41	-49	-57	-65	-73
A	-16	-10	-3	-4	-12	-20	-28	-36	-44	-52	-60
W	-24	-18	-11	-6	-7	-15	-5	-13	-21	-29	-37
H	-32	-14	-18	-13	-8	-9	-13	-7	-3	-11	-19
E	-40	-22	-8	-16	-16	-9	-12	-15	-7	3	-5
A	-48	-30	-16	-3	-11	-11	-12	-12	-15	-5	2
E	-56	-38	-24	-11	-6	-12	-14	-15	-12	-9	1

```
HEAGAWGHE-E
-PA--W-HEAE
```

In reality, our sequences are much longer. Included below is the amino acid sequence of the Northern Spotted Owl:

```
KVRSFEKTPSDDSQHINKDQAEEVTSSNKEIILHKDEAVXRGEKTDLMGBRQALE
KDANDMKTQDSKAHQNNLKQLCRICGVSFKTDHYKRTHPVHGPVDDETLWLL
RKKEKKATSWPDLIAKVFKIDVRGDVDTIHPTQFCHNCWSIIHRKFSNTPCEVYF
PRNSTMEWQPHSPNCDVCRTTSRGVKRKRQPPSVQHGKRVKTMAERARINRGV
KNQAQINNKNLMKELVNCKNIHLSTKLLAVDYPEDFIKSISCQICEHILADPVETT
CRHLFCRTCILKCIKVMGSYCPSCWYPCFPTDLVTPVKSFLNVLDSLGIRCPVKEC
DEEILHGKYGQHLSSHKEMKDRELYCHINKGGRPRQHLLSLTRRAQKHRLRELK
RQVKAFAEKEEGGDIKAVCMTLFLLALRAKNEHRQADELEAIMQGRGSGLHPA
VCLAIRVNTFLSCSQYHKMYRTVKAVTGRQIFQPLHALRTAEKALLPGYHPFEW
KPPLKNVSTNTEVGIIDGLSGLPLSIDDYPVDTIAKRFRYDAALVCALKDMEEEIL
EGMKAKNLDDYLNGPFTVVVKESCDGMGDVSEKHGSGPAVPEKAVRFSFTVM
NIAIALGKESKRIFEEVKPNSELCCKPLCLMLADESDHETLTAILSPLIAEREAMKN
SELLLEMGGILRTFKFIFRGTGYDEKLVREVEGLEASGSTYICTLCDATRLEASQN
LVFHSITRSHAENLERYEIWRSNPYHESADELRDRVKGVSAKPFIETVPSIDALHC
```

DIGNATEFYRIFQMEIGEVYKNPDVSKEERKRWQLTLDKHLRKKMNLKPMMRM
SGNFARKLMSKETVEAVCELIKCEERHEALKELMDLYLKMKPVWRSSCPAKECP
ELLCQYSYNSQRFAELLSTKFKYRYEGKITNYFHKTLAHVPEIIERDGSIGAWASE
GNESGNKLFRRFRKMNARQSKCYEMEDVL.

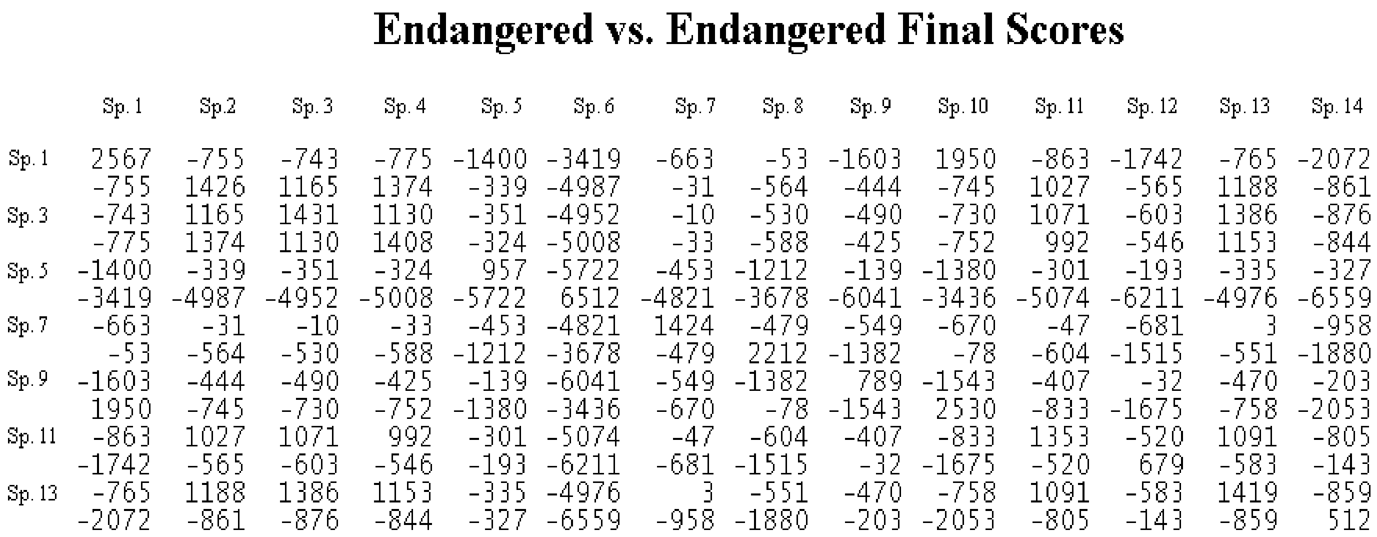
Using the chart on the previous page, the top and left rows are just gap penalties being sequentially added. The final alignment score is located in the bottom right corner of the matrix. A perfect score for two identical sequences cannot be defined, because a longer perfect alignment will have a much higher score than a shorter perfect alignment. The scoring method used in the program is included below:

```
public static int ScoringMethod(int[] geneSeq1, int[] geneSeq2)
  {
    int i;
    int j;
    int a,b,c;
    int max;
    final int penalty=-8;
    int[][]scores=new int[geneSeq1.length+1][geneSeq2.length+1];

    scores[0][0]=0;

    for(i=1; i<=geneSeq1.length; ++i)
    {
      scores[i][0]=penalty+scores[i-1][0];
    }
    for(j=1; j<=geneSeq2.length; ++j)
    {
      scores[0][j]=penalty+scores[0][j-1];
    }
    for(i=1; i<= geneSeq1.length; ++i)
    {
      for(j=1; j<=geneSeq2.length; ++j)
      {
        a=scores[i-1][j]+penalty;
        b=scores[i][j-1]+penalty;
        c=scores[i-1][j-1]+BLOSUM50[geneSeq1[i-1]][geneSeq2[j-1]];
        max=a;
        if(b>max)
        {
          max=b;
```

```
                }
                if(c>max)
                {
                    max=c;
                }
                scores[i][j]=max;
            }
        }
```

The algorithm can be found in nested for loops, allowing it to run through the 784 (28^2) necessary alignments. This number comes from the twenty-eight species being compared with each other. The method then returns “max”, which is the highest score out of the three potential scores possible in the algorithm.

Results



Endangered vs. Endangered Final Scores

	Sp. 1	Sp.2	Sp. 3	Sp. 4	Sp. 5	Sp. 6	Sp. 7	Sp. 8	Sp. 9	Sp. 10	Sp. 11	Sp. 12	Sp. 13	Sp. 14
Sp. 1	2567	-755	-743	-775	-1400	-3419	-663	-53	-1603	1950	-863	-1742	-765	-2072
	-755	1426	1165	1374	-339	-4987	-31	-564	-444	-745	1027	-565	1188	-861
Sp. 3	-743	1165	1431	1130	-351	-4952	-10	-530	-490	-730	1071	-603	1386	-876
	-775	1374	1130	1408	-324	-5008	-33	-588	-425	-752	992	-546	1153	-844
Sp. 5	-1400	-339	-351	-324	957	-5722	-453	-1212	-139	-1380	-301	-193	-335	-327
	-3419	-4987	-4952	-5008	-5722	6512	-4821	-3678	-6041	-3436	-5074	-6211	-4976	-6559
Sp. 7	-663	-31	-10	-33	-453	-4821	1424	-479	-549	-670	-47	-681	3	-958
	-53	-564	-530	-588	-1212	-3678	-479	2212	-1382	-78	-604	-1515	-551	-1880
Sp. 9	-1603	-444	-490	-425	-139	-6041	-549	-1382	789	-1543	-407	-32	-470	-203
	1950	-745	-730	-752	-1380	-3436	-670	-78	-1543	2530	-833	-1675	-758	-2053
Sp. 11	-863	1027	1071	992	-301	-5074	-47	-604	-407	-833	1353	-520	1091	-805
	-1742	-565	-603	-546	-193	-6211	-681	-1515	-32	-1675	-520	679	-583	-143
Sp. 13	-765	1188	1386	1153	-335	-4976	3	-551	-470	-758	1091	-583	1419	-859
	-2072	-861	-876	-844	-327	-6559	-958	-1880	-203	-2053	-805	-143	-859	512

Endangered vs. Non-Endangered Final Scores

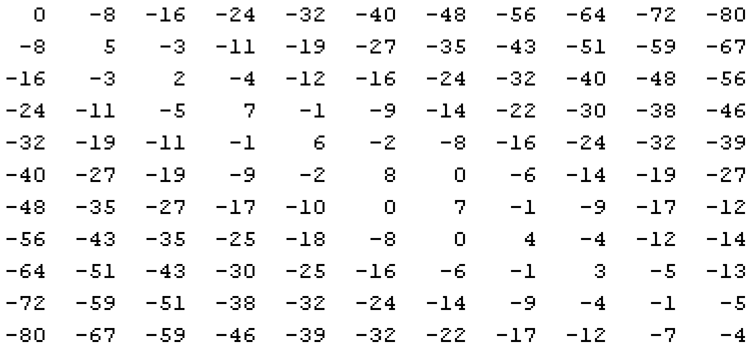
	Sp. 1	Sp.2	Sp. 3	Sp. 4	Sp. 5	Sp. 6	Sp. 7	Sp. 8	Sp. 9	Sp. 10	Sp. 11	Sp. 12	Sp. 13	Sp. 14
Sp. 1	-748	-1385	-849	-805	-1152	2261	-747	-46	-1779	2242	-78	-287	-755	-1534
	-81	-297	1133	-49	-205	-769	1174	-83	-612	-763	-518	-455	1162	-422
Sp. 3	-93	-319	1265	-96	-217	-758	1417	-50	-653	-751	-508	-430	1399	-450
	-78	-289	1094	-46	-194	-798	1135	-84	-602	-786	-543	-468	1127	-404
Sp. 5	-439	-51	-309	-365	-152	-1408	-351	-292	-151	-1415	-1194	-978	-354	-38
	-4804	-5695	-5065	-4857	-5291	-3385	-4961	-5006	-6226	-3390	-3711	-3974	-4958	-5882
Sp. 7	-14	-361	-41	-70	-248	-695	-6	-39	-735	-677	-459	-329	-4	-508
	-535	-1183	-610	-598	-968	-66	-522	-647	-1584	-73	1221	-118	-525	-1317
Sp. 9	-557	-164	-423	-537	-310	-1605	-485	-431	-80	-1593	-1369	-1182	-486	-76
	-696	-1378	-821	-796	-1124	1958	-736	-137	-1759	1974	-66	-262	-737	-1508
Sp. 11	-111	-240	1104	-106	-151	-866	1063	-21	-552	-839	-583	-519	1072	-390
	-673	-198	-523	-657	-411	-1744	-603	-535	-39	-1745	-1526	-1334	-608	-136
Sp. 13	-77	-302	1294	-93	-203	-780	1391	-54	-631	-766	-529	-434	1377	-432
	-903	-389	-822	-880	-570	-2069	-876	-799	-105	-2068	-1873	-1571	-874	-314

Non-Endangered vs. Non-Endangered Final Scores

	Sp. 1	Sp.2	Sp. 3	Sp. 4	Sp. 5	Sp. 6	Sp. 7	Sp. 8	Sp. 9	Sp. 10	Sp. 11	Sp. 12	Sp. 13	Sp. 14
Sp. 1	1500	-374	-114	-76	-233	-745	-94	-89	-713	-730	-504	-377	-91	-487
	-374	983	-256	-372	-130	-1387	-319	-316	-216	-1400	-1171	-923	-311	-2
Sp. 3	-114	-256	1366	-74	-148	-859	1278	-50	-572	-823	-557	-487	1252	-395
	-76	-372	-74	1462	-163	-784	-95	-81	-602	-823	-620	-394	-95	-476
Sp. 5	-233	-130	-148	-163	1206	-1152	-218	-176	-381	-1141	-931	-670	-212	-196
	-745	-1387	-859	-784	-1152	2557	-765	2	-1772	2330	-89	-294	-759	-1538
Sp. 7	-94	-319	1278	-95	-218	-765	1432	-48	-647	-757	-512	-434	1404	-452
	-89	-316	-50	-81	-176	2	-48	1434	-572	-24	-633	-490	-53	-437
Sp. 9	-713	-216	-572	-602	-381	-1772	-647	-572	676	-1773	-1571	-1317	-658	-118
	-730	-1400	-823	-823	-1141	2330	-757	-24	-1773	2557	-91	-282	-755	-1530
Sp. 11	-504	-1171	-557	-620	-931	-89	-512	-633	-1571	-91	2161	-116	-498	-1269
	-377	-923	-487	-394	-670	-294	-434	-490	-1317	-282	-116	1973	-431	-1074
Sp. 13	-91	-311	1252	-95	-212	-759	1404	-53	-658	-755	-498	-431	1432	-454
	-487	-2	-395	-476	-196	-1538	-452	-437	-118	-1530	-1269	-1074	-454	895

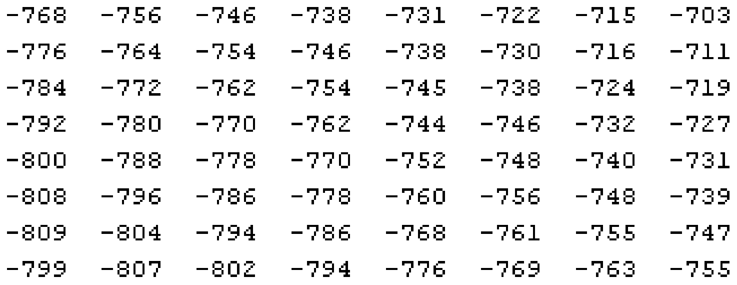
Results cont.

The previous three graphs are the final scores of each individual alignment. Keep in mind that these numbers are the final scores, and not the scoring matrices themselves. A scoring matrix would be several hundred characters long, and would not fit in our results. The first eleven rows and columns of species 1 compared with species 2 are show below:



```
   0   -8  -16  -24  -32  -40  -48  -56  -64  -72  -80
  -8    5   -3  -11  -19  -27  -35  -43  -51  -59  -67
 -16   -3    2   -4  -12  -16  -24  -32  -40  -48  -56
 -24  -11   -5    7   -1   -9  -14  -22  -30  -38  -46
 -32  -19  -11   -1    6   -2   -8  -16  -24  -32  -39
 -40  -27  -19   -9   -2    8    0   -6  -14  -19  -27
 -48  -35  -27  -17  -10    0    7   -1   -9  -17  -12
 -56  -43  -35  -25  -18   -8    0    4   -4  -12  -14
 -64  -51  -43  -30  -25  -16   -6   -1    3   -5  -13
 -72  -59  -51  -38  -32  -24  -14   -9   -4   -1   -5
 -80  -67  -59  -46  -39  -32  -22  -17  -12   -7   -4
```

The last eight rows and columns are show below:



```
 -768  -756  -746  -738  -731  -722  -715  -703
 -776  -764  -754  -746  -738  -730  -716  -711
 -784  -772  -762  -754  -745  -738  -724  -719
 -792  -780  -770  -762  -744  -746  -732  -727
 -800  -788  -778  -770  -752  -748  -740  -731
 -808  -796  -786  -778  -760  -756  -748  -739
 -809  -804  -794  -786  -768  -761  -755  -747
 -799  -807  -802  -794  -776  -769  -763  -755
```

The first partial matrix shows the cumulative penalty running on the first row and column, with the starting zero in the upper left corner. The second partial matrix contains the final score, 755, which is then stored into the final score matrix, located on the previous pages under Sp.1 vs. Sp.2.

Only three final scores charts exist, since endangered vs. non-endangered and non-endangered vs. endangered are identical. One can see the "perfect" scores running diagonally through each of the charts. Interpreting the vast amount of numbers is difficult, so averages of each of the matrices were taken in order to determine which matrix was better aligned. The perfect scores were not taken into account, as comparing animals to themselves plays no significance in our final averages:

Average Non-Endangered vs. Non-Endangered Score = -4202
Average Endangered vs. Endangered Score = -324
Average Non-Endangered vs. Endangered Score = -1877

Taking these rough averages as fact, one can determine that a genetic similarity exists. The greater the score in global alignment, the higher the similarity in our sequences. When aligning massive sequences, negative numbers are common in final scores. This does not mean that the alignment was “bad”, or “un-similar”. This is the very reason we used a control group. From our results, endangered animals in New Mexico have a higher genetic similarity than non-endangered species.

Conclusion

From our results as of now we can imply that our experiment was successful in proving our hypothesis. Our rough averages show we do have a better alignment in our endangered animals; however, this method of calculation is not statistically accurate. Our group intends to perform several statistical tests before the final presentations in Los Alamos. Also, our project results are limited by our choice of animals. A wider range of endangered animals (not necessarily a higher number, too many sequences makes global alignment obsolete) and a more precisely selected control group would reduce our chance of error. Regardless, the results received from the experiment were parallel to our hypothesis, and prove that a genetic similarity does in fact exist in New Mexican endangered animals.

Program

This is the complete program, excluding the twenty-eight text files used to store the genetic sequences.

```
package multialign;

import java.io.*;
import java.lang.Integer.*;


public class Main
{
   private static int SIZE=28;

   // Our coded scheme for the amino acids
   private static final int
            A=0,
            R=1,
            N=2,
            D=3,
            C=4,
            Q=5,
            E=6,
            G=7,
            H=8,
            I=9,
            L=10,
            K=11,
            M=12,
            F=13,
            P=14,
            S=15,
            T=16,
            W=17,
            Y=18,
            V=19;

   private static final int[][] BLOSUM50=
      {
       /*A*/  {5,-2,-1,-2,-1,-1,-1,0,-2,-1,
            -2,-1,-1,-3,-1,1,0,-3,-2,0},
      /*R*/  {-2,7,-1,-2,-4,1,0,-3,0,-4,
            -3,3,-2,-3,-3,-1,-1,-3,-1,-3},
      /*N*/  {-1,-1,7,2,-2,0,0,0,1,-3,
            -4,0,-2,-4,-2,1,0,-4,-2,-3},
      /*D*/  {-2,-2,2,8,-4,0,2,-1,-1,-4,
            -4,-1,-4,-5,-1,0,-1,-5,-3,-4},
      /*C*/  {-1,-4,-2,-4,13,-3,-3,-3,-3,-2,
             -2,-3,-2,-2,-4,-1,-1,-5,-3,-1},
      /*Q*/  {-1,1,0,0,-3,7,2,-2,1,-3,
            -2,2,0,-4,-1,0,-1,-1,-1,-3},
      /*E*/  {-1,0,0,2,-3,2,6,-3,0,-4,
            -3,1,-2,-3,-1,-1,-1,-3,-2,-3},
      /*G*/  {0,-3,0,-1,-3,-2,-3,8,-2,-4,
            -4,-2,-3,-4,-2,0,-2,-3,-3,-4},
      /*H*/  {-2,0,1,-1,-3,1,0,-2,10,-4,
            -3,0,-1,-1,-2,-1,-2,-3,2,-4},
      /*I*/  {-1,-4,-3,-4,-2,-3,-4,-4,-4,5,
            2,-3,2,0,-3,-3,-1,-3,-1,4},
      /*L*/  {-2,-3,-4,-4,-2,-2,-3,-4,-3,2,
            5,-3,3,1,-4,-3,-1,-2,-1,1},
      /*K*/  {-1,3,0,-1,-3,2,1,-2,0,-3,
            -3,6,-2,-4,-1,0,-1,-3,-2,-3},
      /*M*/  {-1,-2,-2,-4,-2,0,-2,-3,-1,2,
            3,-2,7,0,-3,-2,-1,-1,0,1},
      /*F*/  {-3,-3,-4,-5,-2,-4,-3,-4,-1,0,
            1,-4,0,8,-4,-3,-2,1,4,-1},
      /*P*/  {-1,-3,-2,-1,-4,-1,-1,-2,-2,-3,
            -4,-1,-3,-4,10,-1,-1,-4,-3,-3},
      /*S*/  {1,-1,1,0,-1,0,-1,0,-1,-3,
            -3,0,-2,-3,-1,5,2,-4,-2,-2},
      /*T*/  {0,-1,0,-1,-1,-1,-1,-2,-2,-1,
            -1,-1,-1,-2,-1,2,5,-3,-2,0},
      /*W*/  {-3,-3,-4,-5,-5,-1,-3,-3,-3,-3,
            -2,-3,-1,1,-4,-4,-3,15,2,-3},
      /*Y*/  {-2,-1,-2,-3,-3,-1,-2,-3,2,-1,
            -1,-2,0,4,-3,-2,-2,2,8,-1},
      /*V*/  {0,-3,-3,-4,-1,-3,-3,-4,-4,4,
            1,-3,1,-1,-3,-2,0,-3,-1,5}
     };

/*this method imports the raw data from a text
 *document*/

public static String ReadSpecies(String filename)
{
   String line = "";

   try
   {
      FileReader fileReader = new FileReader(filename);
      BufferedReader reader = new BufferedReader(fileReader);
      line = reader.readLine();
      reader.close();
   }
   catch(Exception ex)
   {
      ex.printStackTrace ();
   }

   return line;

}

public static int[] SwitchProtein(String geneSeqStr)
{
   int i;
   int j;

   int [] geneSeq = new int[geneSeqStr.length ()];

   for(i=0; i < geneSeqStr.length(); ++i)
   {
      switch (geneSeqStr.charAt (i))
      {
         case 'A': geneSeq[i]=A;
            break;
         case 'R': geneSeq[i]=R;
            break;
         case 'N': geneSeq[i]=N;
            break;
         case 'D': geneSeq[i]=D;
            break;
         case 'C': geneSeq[i]=C;
            break;
         case 'Q': geneSeq[i]=Q;
            break;
         case 'E': geneSeq[i]=E;
            break;
         case 'G': geneSeq[i]=G;
            break;
         case 'H': geneSeq[i]=H;
```

```
                break;
            case 'I': geneSeq[i]=I;
                break;
            case 'L': geneSeq[i]=L;
                break;
            case 'K': geneSeq[i]=K;
                break;
            case 'M': geneSeq[i]=M;
                break;
            case 'F': geneSeq[i]=F;
                break;
            case 'P': geneSeq[i]=P;
                break;
            case 'S': geneSeq[i]=S;
                break;
            case 'T': geneSeq[i]=T;
                break;
            case 'W': geneSeq[i]=W;
                break;
            case 'Y': geneSeq[i]=Y;
                break;
            case 'V': geneSeq[i]=V;
                break;
        }
    }

    return geneSeq;
}


/*
 *this is the scoring portion of the program
 *where the scores are calculated by comparing
 *two sequences, geneSeq1 and geneSeq2, and
 *inputting the results into a value, max
 **/
public static int ScoringMethod(int[] geneSeq1, int[] geneSeq2)
{
    int i;
    int j;
    int a,b,c;
    int max;
    final int penalty=-8;
    int[][]scores=new int[geneSeq1.length+1][geneSeq2.length+1];

    scores[0][0]=0;
    for(i=1; i<=geneSeq1.length; ++i)
    {
      scores[i][0]=penalty+scores[i-1][0];
    }
    for(j=1; j<=geneSeq2.length; ++j)
    {
      scores[0][j]=penalty+scores[0][j-1];
    }
    for(i=1; i<= geneSeq1.length; ++i)
    {
      for(j=1; j<=geneSeq2.length; ++j)
      {
        a=scores[i-1][j]+penalty;
        b=scores[i][j-1]+penalty;
        c=scores[i-1][j-1]+BLOSUM50[geneSeq1[i-1]][geneSeq2[j-1]];
        max=a;
        if(b>max)
        {
          max=b;
        }
        if(c>max)
        {
          max=c;
        }
        scores[i][j]=max;
      }
    }

   System.out.println();
    System.out.println();
    for( i=0; i<=geneSeq1.length; ++i)
    {
      for( j=0; j<= geneSeq2.length; ++j)
      {
       System.out.printf("%4d ", scores[i][j]);
      }
      System.out.println();
    }

    return scores[ geneSeq1.length][geneSeq2.length];
  }
```

```
public static void main(String[] args)
   throws IOException, FileNotFoundException
{
   int i;
   int j;
   String[]SpeciesStr = new String[SIZE+1];
   int[][] finalScores = new int[SIZE+1][SIZE+1];
   int[][]Sequence =new int[SIZE+1][];
   PrintWriter out =
      new PrintWriter( new FileOutputStream("Results.txt"), true);

   /*
   *this for loop is the method that reads the raw data
   *from the text files
   */

   for(i=1; i<=SIZE;++i)
   {
      String speciesFile = "species" +
         new Integer(i).toString() + ".txt";

      SpeciesStr[i]=ReadSpecies(speciesFile);
      //System.out.println(SpeciesStr[i]);
   }

   /*this for loop is for the switch statement method
   *and creates the array that contains the integer
   *values
   */

   for(i=1; i<=SIZE; ++i)
   {
      Sequence[i]=SwitchProtein(SpeciesStr[i]);
      for( j=0; j<Sequence[i].length; ++j )
      {
         //System.out.printf("%3d",Sequence[i][j]);
      }
      //System.out.println();
   }

   /*this for loop is for the actual experiment
     and creates an array to hold the results*/
```

```
for(i=1; i<SIZE+1; ++i)
{
   for(j=1; j<SIZE+1; ++j)
   {
      finalScores[i][j]=ScoringMethod(Sequence[i], Sequence[j]);
      out.printf( "%6d",finalScores[i][j]);
      if( j==14 )
      {
         out.printf("    ");
      }
   }
   out.println();
   if( i==14 )
   {
      out.println();
      out.println();
   }
}
out.close();

int TotalNonNon=0;
int TotalNonEnd=0;
int TotalEndEnd=0;

for(i=15; i<=SIZE; ++i)
{
   for(j=15; j<=SIZE; ++j)
   {
      TotalNonEnd += finalScores[i][j];
   }
}

for(i=1; i<=14; ++i)
{
   for(j=1; j<=14; ++j)
   {
      TotalEndEnd += finalScores[i][j];
   }
}

for(i=15; i<=28; ++i)
{
   for(j=15; j<=28; ++j)
   {
      TotalNonNon += finalScores[i][j] - finalScores[i][i];
   }
```

```
        }

        System.out.println("Average Non vs. Non Score = " +
                TotalNonNon/91);
        System.out.println("Average End vs. End Score = " +
                TotalEndEnd/196);
        System.out.println("Average Non vs. End Score = " +
                TotalNonEnd/91);
    }
}
```

Most Significant Original Achievement

Our most significant original achievement is finding that the fourteen endangered species native to New Mexico have a higher genetic similarity than our control set of species. Statistical tests are still necessary to state our findings as fact, but the average scores were convincing enough to say our hypothesis was correct, and our experiment was a success.

Acknowledgements

We would like to thank Mr. Downs for his constant devotion to our project. Without his guidance, there would not have been a program or a report. His contributions to our idea and the books he provided were the reason our project succeeded. Thank you Mr. Downs, this project would have never happened without you.

Appendix

Endangered Species List:

1. Lesser Long Nosed Bat
2. Whooping Crane
3. Bald Eagle
4. Willow Flycatcher
5. Jaguar
6. Northern Spotted Owl
7. Animas Ridge-Nosed Rattlesnake
8. Arkansas River Shiner
9. Beautiful Shiner
10. Spikedace
11. Alamosa Springsnail
12. Razerback Sucker
13. Least Tern
14. Gila Topminnow

Non-Endangered Species List:

15. Pronghorn Antelope
16. Javelina
17. Black-Throated Sparrow
18. Pike
19. White-Tailed Deer
20. Ringtail
21. Brown Towhee
22. Elk
23. Wild Turkey
24. Black Bear
25. Hairy Woodpecker
26. Pine Marten
27. Blue Grouse
28. Bighorn Sheep

Bibliography

--"Bioinformatics." *Wikipedia, The Free Encyclopedia*. 2 Apr 2007, 20:49 UTC. Wikimedia Foundation, Inc. 3 Apr 2007

--Durbin, R.; Eddy, S.; Krogh, A.;Mitchison, G.; 1998. *Biological Sequence Analysis.* Cambridge, United Kingdom. Published by the Press Syndicate of the University of Cambridge.

--Jean-Michel Claverie, PhD; Cedric Notredame, PhD. 2003. *Bioinformatics For Dummies.* New York: Wiley Publishing, Inc.