Determining the number and Type of AT Domains in Random PKS Amino Acid Sequence

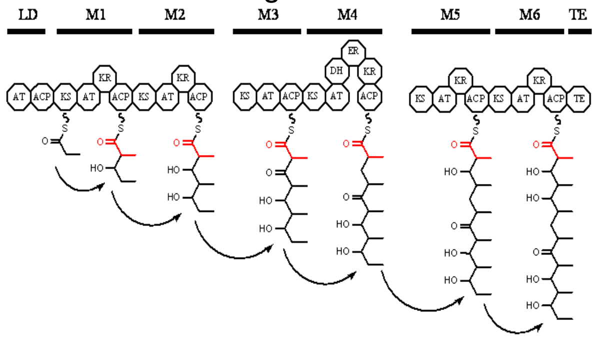
Neil Klenk (nlk322), Lee Sugarek (lms3748), Kyoung Won Oe(ko4696)

The main goal for this presentation was to determine the number and type of AT domains in a polyketide synthase (PKS) amino acid sequence. PKS's are of serious interest within the context of drug discovery, and the AT domain is a crucial part of the synthase.

Background

PKS's function much like an assemble line. They are made up of many modules, and each module is made up of many domains as can be seen in figure 1.

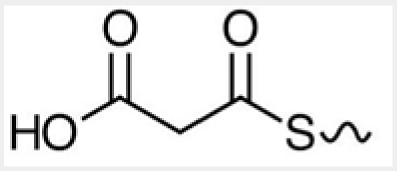
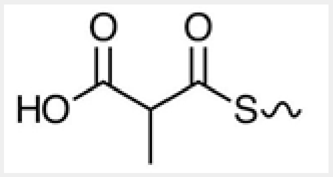
Figure 1.



The regions labeled with an M are modules, while the hexagons located inside are different domains

Everytime the growing carbon chain is being passed to a new module, the chain is extended by 2 carbons. There are different side groups that can be added onto the 2 carbon extension, and that is determined by the presence of different domains in the modules. The AT domain is of particular interest because it is responsible for the type of extender unit that is used. There are two basic types of AT domains. One creates a malonyl-Coa extender unit, and the other creates a Methylmalonyl-Coa extender unit. The way that the two are differentiated in a silico (computational) analysis is by a motif. Motifs are highly conserved series of amino acids that act as identifiers for particular proteins. Essentially this particular series of amino acids are crucial to this protein to function, and therefore if it is not present then the protein isn't there either. The different motifs and resulting structures can be seen in figure 2.

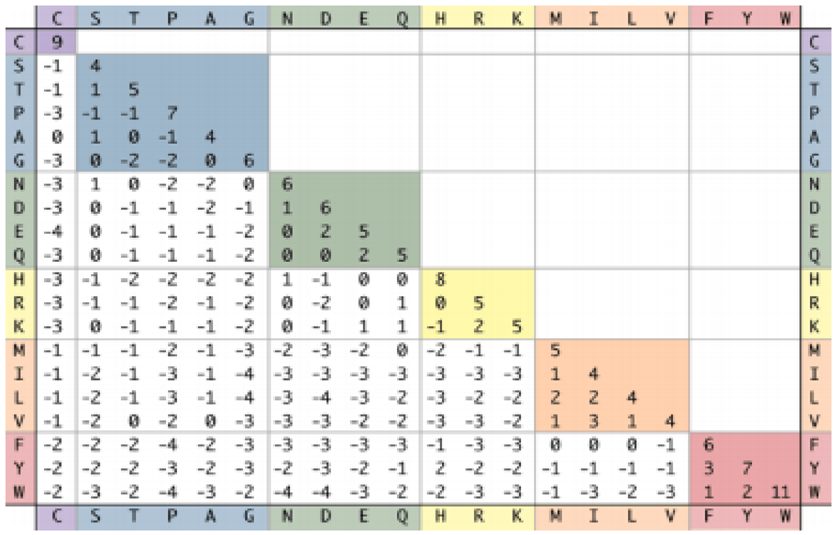
Figure 2: Extender unit, Structure, and Motif for the 2 AT domain types

Extender Unit	**Structure**	**Motif**
Malonyl-CoA		HAFH
Methylmalonyl-CoA		YASH

Blossum62 Matrix

One prevailing theme in biology is that there is no such thing as always. Nature is continually changing in ways that we cannot imagine, and so when doing scientific inquiries into realms such as biology, this needs to be accounted for. The Blossum62 matrix is a well established way to account for variance when looking for matches in between amino acid sequences (commonly call sequence alignment), and can be seen in figure 3.

Figure 3. Blossum62 Matrix



	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	
C	9																				C
S	-1	4																			S
T	-1	1	5																		T
P	-3	-1	-1	7																	P
A	0	1	0	-1	4																A
G	-3	0	-2	-2	0	6															G
N	-3	1	0	-2	-2	0	6														N
D	-3	0	-1	-1	-2	-1	1	6													D
E	-4	0	-1	-1	-1	-2	0	2	5												E
Q	-3	0	-1	-1	-1	-2	0	0	2	5											Q
H	-3	-1	-2	-2	-2	-2	1	-1	0	0	8										H
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	5									R
K	-3	0	-1	-1	-1	-2	0	-1	1	1	-1	2	5								K
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	5							M
I	-1	-2	-1	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	4						I
L	-1	-2	-1	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	4					L
V	-1	-2	0	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	4				V
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	6			F
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	7		Y
W	-2	-3	-2	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	11	W
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W	

http://www.life.umd.edu/classroom/BSCI410/images/BLOSUM.jpg

This matrix functions by taking the amino acid from the query sequence (in our case the motif), and the strand against which the comparison is being done. The two letters are matched and a score is given, ranging from 11 to -4. This score is the result of numerous factors including difference in thermodynamic stability of the two amino acids, the differences in the sterics, differences in regions of electro negativity or positivity, etc. The greater the difference, the less likely it is for our sought protein to fold properly. However, some amino acids are quite similar and will not affect the structure much resulting in a score that is still positive. This matrix functions to allow for a protein motif to be recognized even if the query sequence and the comparison strand don't match perfectly, as long as whatever mutation does not harm the resultant protein to the point that it cannot perform its function. See figure 4 for an example calculation (notice the negative score for the Q and Y pair).

Figure 4. Example Blossum63 Matrix Scoring

M scores	8	-1	6	4	4	5
query	H	Q	G	L	L	T
database	H	Y	G	L	L	T

S = 8 - 1 + 6 + 4 + 4 + 5 = 25

Test Case

The sequence that was used for our test case was pulled from Clustermine360, a database for microbial PKS biosynthesis. It contained a malonyl-CoA extender unit, and the motif for the methylmalonyl-CoA extender unit was appended to the beginning of the sequence. A basic header label was added as well to make it more closely resemble a fasta file that would be located on NCBI, the website for The National Center of Biotechnology.

>Seq1

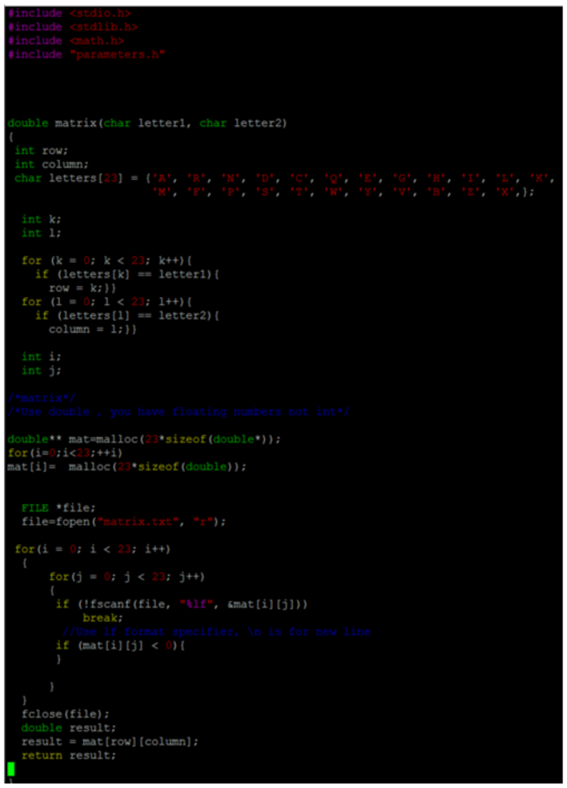
**YASHLFTGQGSQRIGAGRELAARFPVFAEALDTVLSAFDPELQRPLREILRRTTDGTTPDETA
AALLDDTAYAQPAIFAVEVALYRLVTSLGVTPGHLLGHSVGEIAAAHVAGVLALDDAATLVAA
RGRLMAALPTGGAMVAVQATEAEVEALLDGYRDRVSVAAVNGPSAVVLSGADEAVTAVAG
LLADQGRRTRRLRVSHAFHSPLMEPMLDEFRAVVEGLDLQAPLIPIVSDVTGELATVAQLTSA
DYWVEHVRSAVRFADGVDWLARHDVTAFLELGPDGPLSAMTRDCLDAADPGAPASVAVPA
LR**

Game Plan

The game plan for this project is to scan the sequence for alignments to meet a threshold score T. This will be done calculating the score by using a Blossum matrix. The code will then report the alignments that exceeds the cutoff score S.

Code

makematrix.c



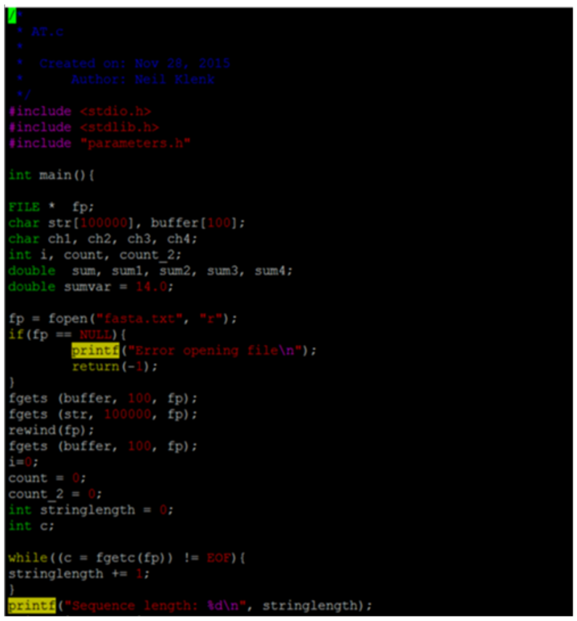
```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "parameters.h"



double matrix(char letter1, char letter2)
{
 int row;
 int column;
 char letters[23] = {'A', 'R', 'N', 'D', 'C', 'Q', 'E', 'G', 'H', 'I', 'L', 'K',
                     'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', 'B', 'Z', 'X',};

  int k;
  int l;

  for (k = 0; k < 23; k++){
    if (letters[k] == letter1){
      row = k;}}
  for (l = 0; l < 23; l++){
    if (letters[l] == letter2){
      column = l;}}

  int i;
  int j;

/*matrix*/
/*Use double , you have floating numbers not int*/

double** mat=malloc(23*sizeof(double*));
for(i=0;i<23;++i)
mat[i]=  malloc(23*sizeof(double));


  FILE *file;
  file=fopen("matrix.txt", "r");

 for(i = 0; i < 23; i++)
  {
      for(j = 0; j < 23; j++)
      {
       if (!fscanf(file, "%lf", &mat[i][j]))
           break;
        //Use lf format specifier, \n is for new line
       if (mat[i][j] < 0){
       }

      }
  }
  fclose(file);
  double result;
  result = mat[row][column];
  return result;
```

Takes the arguments (letter1 and letter2) from the main function, and finds which row and column they reside on. The order that the letters are present in the letters array corresponds to their order in the Blossum62 matrix.

Opens "matrix.txt", a text file pulled from the NCBI database to create the Blossum62 matrix, in order to determine the score for each letter pair

Returns the value that corresponds to that character pair as a double for use in the main program

Compare.c



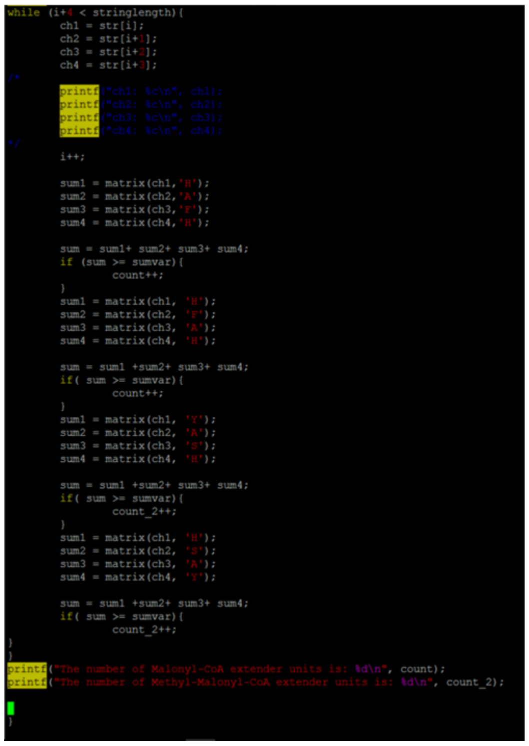
```c
/*
 * AT.c
 *
 *  Created on: Nov 28, 2015
 *      Author: Neil Klenk
 */
#include <stdio.h>
#include <stdlib.h>
#include "parameters.h"

int main(){

FILE *  fp;
char str[100000], buffer[100];
char ch1, ch2, ch3, ch4;
int i, count, count_2;
double  sum, sum1, sum2, sum3, sum4;
double sumvar = 14.0;

fp = fopen("fasta.txt", "r");
if(fp == NULL){
        printf("Error opening file\n");
        return(-1);
}
fgets (buffer, 100, fp);
fgets (str, 100000, fp);
rewind(fp);
fgets (buffer, 100, fp);
i=0;
count = 0;
count_2 = 0;
int stringlength = 0;
int c;

while((c = fgetc(fp)) != EOF){
stringlength += 1;
}
printf("Sequence length: %d\n", stringlength);
```

sumvar is the threshold that we will use to determine if there was a match or not

Opens the designated fasta file (fasta.txt)

fgets to store all of the header info into the buffer
fgets to stores all of the sequence info into the “str” variable
rewind to reset the pointer location
fgets to put pointer at the end of the header

Each time fgetc was able to put a value into c (hadn’t reached the end of the file) increment the length of “stringlength” by 1. This was used to determine the length of our sequence.



```c
while (i+4 < stringlength){
        ch1 = str[i];
        ch2 = str[i+1];
        ch3 = str[i+2];
        ch4 = str[i+3];
/*
        printf("ch1: %c\n", ch1);
        printf("ch2: %c\n", ch2);
        printf("ch3: %c\n", ch3);
        printf("ch4: %c\n", ch4);
*/
        i++;

        sum1 = matrix(ch1,'H');
        sum2 = matrix(ch2,'A');
        sum3 = matrix(ch3,'F');
        sum4 = matrix(ch4,'H');

        sum = sum1+ sum2+ sum3+ sum4;
        if (sum >= sumvar){
                count++;
        }
        sum1 = matrix(ch1, 'H');
        sum2 = matrix(ch2, 'F');
        sum3 = matrix(ch3, 'A');
        sum4 = matrix(ch4, 'H');

        sum = sum1 +sum2+ sum3+ sum4;
        if( sum >= sumvar){
                count++;
        }
        sum1 = matrix(ch1, 'Y');
        sum2 = matrix(ch2, 'A');
        sum3 = matrix(ch3, 'S');
        sum4 = matrix(ch4, 'H');

        sum = sum1 +sum2+ sum3+ sum4;
        if( sum >= sumvar){
                count_2++;
        }
        sum1 = matrix(ch1, 'H');
        sum2 = matrix(ch2, 'S');
        sum3 = matrix(ch3, 'A');
        sum4 = matrix(ch4, 'Y');

        sum = sum1 +sum2+ sum3+ sum4;
        if( sum >= sumvar){
                count_2++;
}
}
printf("The number of Malonyl-CoA extender units is: %d\n", count);
printf("The number of Methyl-Malonyl-CoA extender units is: %d\n", count_2);

}
```

The while condition checks ahead to make sure that there are still at least 4 characters left in str by comparing to overall stringlength

Ch1-4 are assigned to the next 4 values in str

Increments the count variable that is associated with a malonyl-CoA extender unit if the threshold score is met for the “HAFH” motif in either the forward or reverse direction.

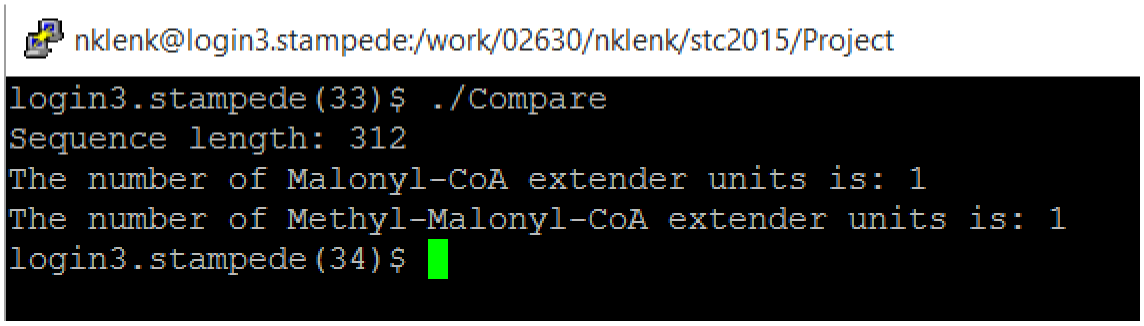
Increments the count_2 variable that is associated with a methylmalonyl-CoA extender unit if the threshold score is met for the “YASH” motif in either the forward or reverse direction.

Prints out the number of each type of extender unit present.

Results

Using a test case that included a header as well as both variants of the motif, we were able to successfully return the correct number and type of each AT domain using a threshold score as low as 14. The scores for the motifs that we are looking for have score as follow if they received a perfect alignment:

- Ideal HAFH score: 26
- Ideal YASH score: 23



nklenk@login3.stampede:/work/02630/nklenk/stc2015/Project

```
login3.stampede(33)$ ./Compare
Sequence length: 312
The number of Malonyl-CoA extender units is: 1
The number of Methyl-Malonyl-CoA extender units is: 1
login3.stampede(34)$ 
```

Future Works

Be able to identify the rest of the domains in order in a PKS

Better alignment scoring rules
-gap penalty of 8 for each gap that exists
-gap length penalty of 1 for each amino acid in the gap

```
M scores    8-1   6  4  4  5           4  5     4  6  5  7-1      4
query       H  Q  G  L  L  T  S  W  V  S  Q  -  S  F  T  P  P  -  I
database    H  Y  G  L  L  T  -  -  -  S  Q  G  S  F  T  P  S  G  I

S = 60 - (3x8) - 5(1) = 31
```