7.36/7.91/20.390/20.490/6.802/6.874
PROBLEM SET 4. Bayesian Networks, Refining Protein Structures in PyRosetta, Mutual information of protein residues (21 Points)

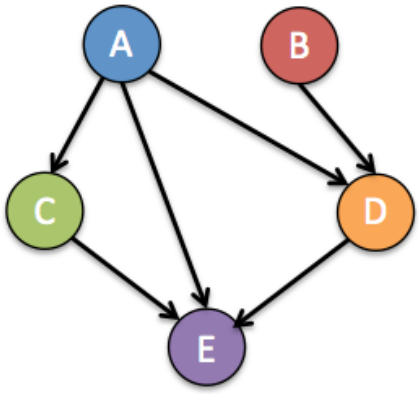
Due: Thursday, April 17th at noon in the dropbox labeled with 7.36/7.91 outside of the Biology Education Office on the ground floor of Building 68.

Python Scripts
All Python scripts must work on athena using /usr/athena/bin/python. You **may not assume availability of any third party modules** unless you are explicitly instructed so. You are advised to test your code on Athena before submitting. Please only modify the code between the indicated bounds, with the exception of adding your name at the top, and remove any print statements that you added before submission.

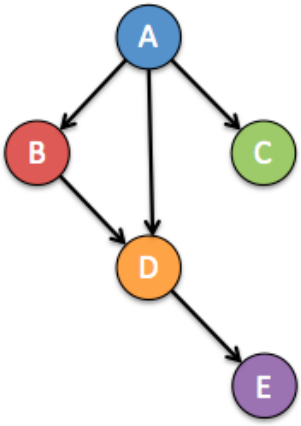
Electronic submissions are subject to the same late homework policy as outlined in the syllabus and submission times are assessed according to the server clock. **Any Python programs you add code to must be submitted electronically, as .py files** on the course website using appropriate filename for the scripts as indicated in the problem set or in the skeleton scripts provided on Stellar. To submit a file electronically:
1. Go to
http://stellar.mit.edu/S/course/7/sp14/7.36/homework/index.html
2. Click on the corresponding problem set.
3. On the Assignment Details page, click the Add Submission link.
4. On the Add Submission page, select the appropriate file on your computer. **Do not use the paste box, do not zip files.**
5. Click Submit when ready.

P1 - Bayesian Networks (7 points)

You are given two different Bayesian network structures 1 and 2, each consisting of 5 binary random variables A, B, C, D, E. Each variable corresponds to a gene, whose expression can be either "ON" or "OFF".



Network 1



Network 2

(A – 2 points) In class, we covered the chain rule of probability for Bayes Nets, which allows us to factor the joint probability over all the variables into terms of conditional probabilities. For each of the following cases, factor P(A,B,C,D,E) according to the independencies specified and give the **minimum** number of parameters required to fully specify the distribution.

(i) A,B,C,D,E are all mutually independent

P(A,B,C,D,E) = P(A)P(B)P(C)P(D)P(E)

5 parameters (probability that each of the 5 genes is ON, independent of others)

(ii) A,B,C,D,E follow the independence assumptions of **Network #1** above

P(A,B,C,D,E) = P(A)P(B)P(C|A)P(D|A,B)P(E|A,C,D)

16 parameters: 1 for P(A), 1 for P(B), 2 for P(C|A), 4 for P(D|A,B), and 8 for P(E|A,C,D)

(iii) A,B,C,D,E follow the independence assumptions of **Network #2** above

P(A,B,C,D,E) = P(A)P(B|A)P(C|A)P(D|A,B)P(E|D)

11 parameters: 1 for P(A), 2 for P(B|A), 2 for P(C|A), 4 for P(D|A,B), 2 for P(E|D)

(iv) no independencies

P(A,B,C,D,E) cannot be simplified

2^5 - 1= 31 parameters (there are 32 combinations of A,B,C,D,E, must sum to 1)

(B – 3 points) Using **Network #2** and the probabilities given below, calculate the probability of the following:

$$P(A = ON) = 0.6$$

$$P(B = ON \mid A) = \begin{cases} 0.1, & A = OFF \\ 0.95, & A = ON \end{cases}$$

$$P(C = ON \mid A) = \begin{cases} 0.8, & A = OFF \\ 0.5, & A = ON \end{cases}$$

$$P(D = ON \mid A, B) = \begin{cases} 0.1 & A = OFF, B = OFF \\ 0.9 & A = ON, B = OFF \\ 0.3 & A = OFF, B = ON \\ 0.95 & A = ON, B = ON \end{cases}$$

$$P(E = ON \mid D) = \begin{cases} 0.8, & D = OFF \\ 0.1, & D = ON \end{cases}$$

(i) P(A=ON, B=ON, C=ON, D=ON, E=ON)

P(A=ON, B=ON, C=ON, D=ON, E=ON)
= P(A=ON)P(B=ON|A=ON)P(C=ON|A=ON)P(D=ON|A=ON,B=ON)P(E=ON|D=ON)
= (0.6)(0.95)(0.5)(0.95)(0.1)
= 0.0271

(ii) P(E = ON | A = ON)

B, D, and E are conditionally independent of C given A, so C drops out. Therefore, we sum over the 4 {B, D} possibilities:

$$\mathbf{P(E = ON \mid A = ON) = \sum_{B,D=\{ON,OFF\}} P(E = ON \mid D)P(D \mid A = ON,B)P(B \mid A = ON)}$$

B	D	P(B|A=ON)	P(D|A=ON,B)	P(E=ON|D)	P(E=ON, B, D|A=ON)
ON	ON	0.95	0.95	0.1	0.09025
ON	OFF	0.95	0.05	0.8	0.038
OFF	ON	0.05	0.9	0.1	0.0045
OFF	OFF	0.05	0.1	0.8	0.004

Summing over the last column, we obtain P(E=ON | A = ON) = 0.13675.

(iii) P(A = ON | E = ON)

By Bayes' rule,

$$P(A = ON \mid E = ON) = \frac{P(E = ON \mid A = ON)P(A = ON)}{P(E = ON)}$$

$$= \frac{P(E = ON \mid A = ON)P(A = ON)}{P(E = ON \mid A = ON)P(A = ON) + P(E = ON \mid A = OFF)P(A = OFF)}$$

We already have P(E=ON | A=ON) from (ii), so we just need P(E=ON | A=OFF):

B	D	P(B|A=OFF)	P(D|A=OFF,B)	P(E=ON|D)	P(E=ON, B, D|A=OFF)
ON	ON	0.1	0.3	0.1	0.003
ON	OFF	0.1	0.7	0.8	0.056
OFF	ON	0.9	0.1	0.1	0.009
OFF	OFF	0.9	0.9	0.8	0.648

Summing over the last column, we obtain P(E=ON | A=OFF) = 0.716. Therefore

$$\mathbf{P(A = ON \mid E = ON) = \frac{(0.13675)(0.6)}{(0.13675)(0.6) + (0.716)(0.4)} = 0.2227}$$

(C – 1 point) For the rest of this problem, you will be using the Python module Pebl (https://code.google.com/p/pebl-project/), which provides an environment for learning the structure of a Bayesian network. Just like PyRosetta, Pebl has been installed on Athena, and we will provide instructions for how to complete this problem on Athena's Dialup Service. You are, of course, free to download Pebl yourself and complete the problem locally.

Log on to Athena's Dialup Service:

```
ssh <your Kerberos username>@athena.dialup.mit.edu
```

Before running any Python scripts, use the following command to add the Athena Pebl module to your `PYTHONPATH`:

```
export PYTHONPATH=/afs/athena/course/20/20.320/pythonlib/lib/python2.7/site-packages/
```

If you forget to do this, you will get
`ImportError: No module named pebl`
when you try to run the provided scripts. Additional information about Pebl and its modules can be found here: https://pythonhosted.org/pebl/apiref.html#apiref.

You will also need to get the .zip containing the files for this problem in the course folder:

```
cd /afs/athena/course/7/7.91/sp_2014
cp bayesNetworks.zip ~
cd ~
unzip bayesNetworks.zip
cd bayesNetworks
```

We have provided you with a small subset of the microarray data that were published in a 2000 paper (Gasch *et al.* - http://www.ncbi.nlm.nih.gov/pubmed/11102521). Gasch *et al*. explored gene expression changes in *S. cerevisiae* in response to a variety of environmental stresses, such as heat shock and oxidative stress. We have given you `geneExprData.txt`, which contains data for 12 of the genes included in the study, observed under these various stress conditions.

We have also provided a simple script `learnNetwork.py` which will learn the network structure that best explains these data, using Pebl's greedy learner algorithm (see https://pythonhosted.org/pebl/learner/greedy.html). From the folder containing `geneExprData.txt`, run the script:

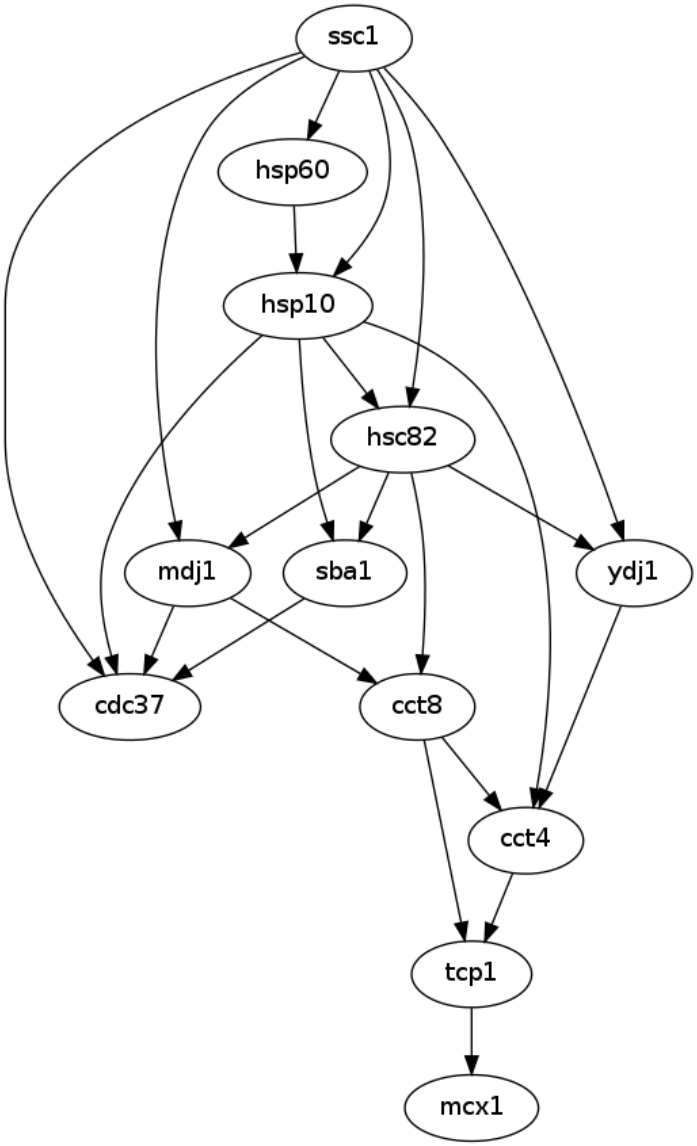
```
python learnNetwork.py geneExprData.txt network1
```

If you've done this correctly, a new folder called `network1` will be created in your current directory on Athena, containing an `.html` file and two more folders `data` and `lib`. If you are at an Athena workstation, you can open and view the `.html` directly. If you've ssh'ed into Athena from your own computer, you will need to copy the entire `outFolderName` to your local computer using scp:

```
<in a new Terminal on your computer, cd into your local computer's directory where you want
to download the files>
scp –r <your Kerberos username>@athena.dialup.mit.edu:~/bayesNetworks/network1 .
```

Now click on the `.html` file to open it. Include a printout of the top scoring network with your write-up or upload a photo of it to the Stellar online dropbox. What is its log score?

Log score = -1966.216



(D – 1 point) We have provided another data file, `exprDataMisTCP1.txt`, which is the same as `geneExprData.txt` except the data for tcp1 has been removed.

(i) Before using Pebl to calculate the network, make a quick guess about how the network might change in response to removing the tcp1 data.

We might assume that we will simply see edges from tcp1's parents going directly to mcx1. In other words, mcx1 expression would depend only on cct8 and cct4.

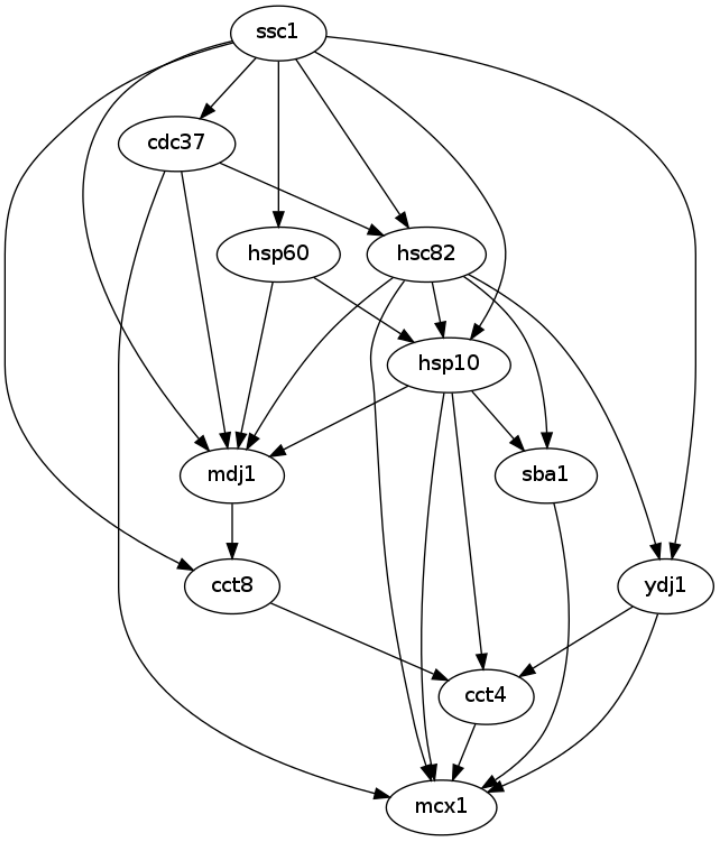
(ii) Now, use the `learnNetwork.py` script to learn a network for `exprDataMisTCP1.txt`:

```
python learnNetwork.py exprDataMisTCP1.txt network2
```

Again, if you ssh'ed into Athena, you will need to copy the `network2` folder locally in order to view it:

```
<in a new Terminal on your computer, cd into your local computer's directory where you want
to download the PDF>
scp –r <your Kerberos username>@athena.dialup.mit.edu:~/bayesNetworks/network2 .
```

Include a printout of the top scoring network with your write-up or upload a photo of it to the Stellar online dropbox. According to this network, which node(s) does the expression of mcx1 depend on? Is this consistent with your guess above? Briefly suggest a reason why you might be observing this network in response to loss of tcp1 data.



When we lose the tcp1 data, we obtain a network in which mcx1 depends on a number of other nodes, not just the parents of tcp1. Losing tcp1, brought together diverse sources of information, makes all the relationships between mcx1 and ancestors of tcp1 more noisy, and makes it harder to tell which are real and which aren't – in this case, it appears that a number of them have some contribution to the value of mcx1.

P2 – Refining Protein Structures in PyRosetta (7 points)

In this problem, you will explore refining protein structures using two methods discussed in class: Energy Minimization and Simulated Annealing. To do this, we will use PyRosetta, an interactive Python-based interface to the powerful Rosetta molecular modeling suite.

When implementing your solutions in the skeleton code provided, we encourage you to look at the solutions from Problem Set 3 Question 3 as well as the PyRosetta documentation: http://graylab.jhu.edu/~sid/pyrosetta/downloads/documentation/PyRosetta_Manual.pdf, particularly Units 2 (Protein Structure in PyRosetta) and 3 (Calculating Energies in PyRosetta), and the Appendix.

In the following, we will provide instructions on how to complete this problem on Athena's Dialup Service, where PyRosetta has previously been installed as a module for another class.

Log onto Athena's Dialup Service, either at a workstation on campus or from a Terminal on your personal machine:

```
ssh <your Kerberos username>@athena.dialup.mit.edu
```

Once logged on, load the PyRosetta module with the following 2 commands:

```
cd /afs/athena/course/20/20.320/PyRosetta

source SetPyRosettaEnvironment.sh
```

If you log onto Athena to complete the problem at a future time, you will have to execute these two commands again, otherwise you will get the following error:

```
ImportError: No module named rosetta
```

Now, head to the course's Athena directory, copy the `pyRosetta_RefiningStructures.zip` folder of files for this problem to your home directory (~), and then unzip it in your home directory with the following commands:

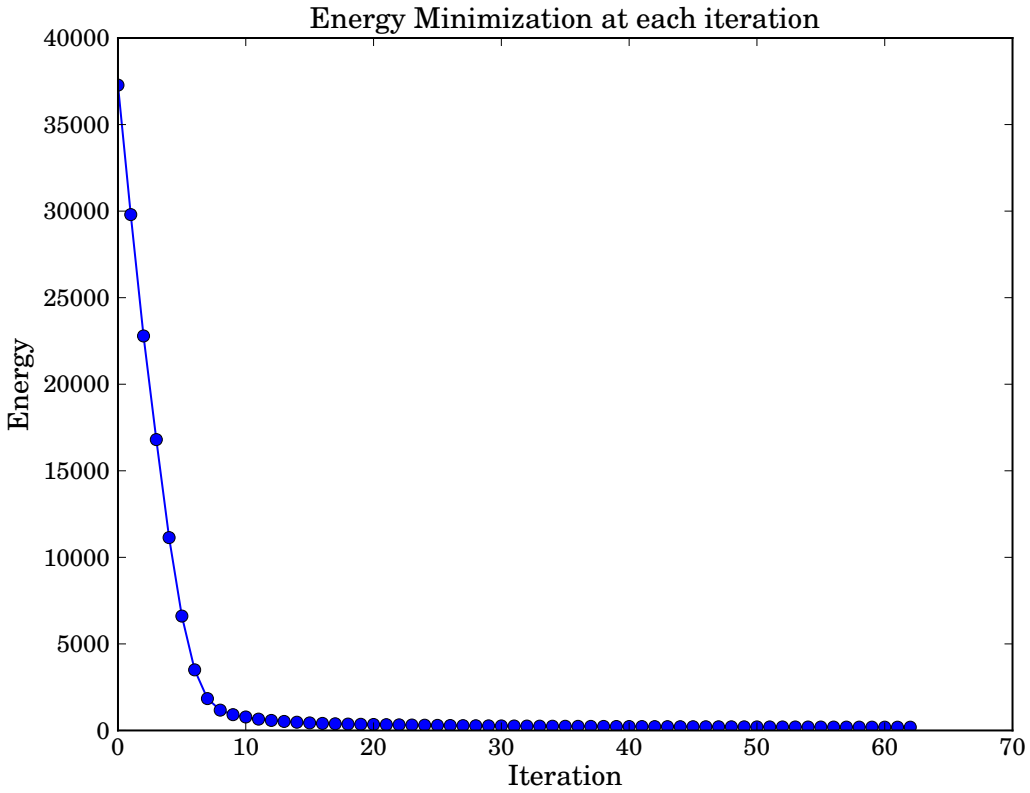
```
cd /afs/athena/course/7/7.91/sp_2014

cp pyRosetta_RefiningStructures.zip ~

cd ~

unzip pyRosetta_RefiningStructures.zip

cd pyRosetta_RefiningStructures
```

You should now be able to edit any of these files. Skeleton code is provided in `pyRosetta_1EK8.py`. The `1EK8.clean.pdb` file is a cleaned PDB file of the *E. coli* ribosome recycling factor, while `1EK8.rotated.pdb` is the same structure but with some phi and/or psi angles rotated. In this problem, you will refine the `1EK8.rotated.pdb` structure using an Energy Minimization approach as well as a Simulated Annealing approach.

(A – 1 point) Complete the `part_a_energy_minimization()` function in `pyRosetta_1EK8.py` to carry out a simple greedy energy minimization algorithm. Rather than computing the gradient of the full energy potential, you should simply calculate the energy of the structure for each of the residue's phi and psi angles changed by ±1 degree, and accept the structure with the lowest energy, continuing until the change in energy is less than 1 (see the code for more specific details).

What are the starting and final energies of the structure? How many iterations until convergence? If implemented correctly, a plot of the energy at each iteration (`energy_minimization_plot.pdf`) will be made; upload it to the Stellar electronic dropbox or include a printout of it with your write-up. Once finished with this, to cut down on run-time for subsequent parts of the question you may want to comment out the lines between `###### PART (A) ########` and `###### END PART (A) ########`.

The starting energy is 37,278.5, and after 61 iterations the final energy is 188.9.



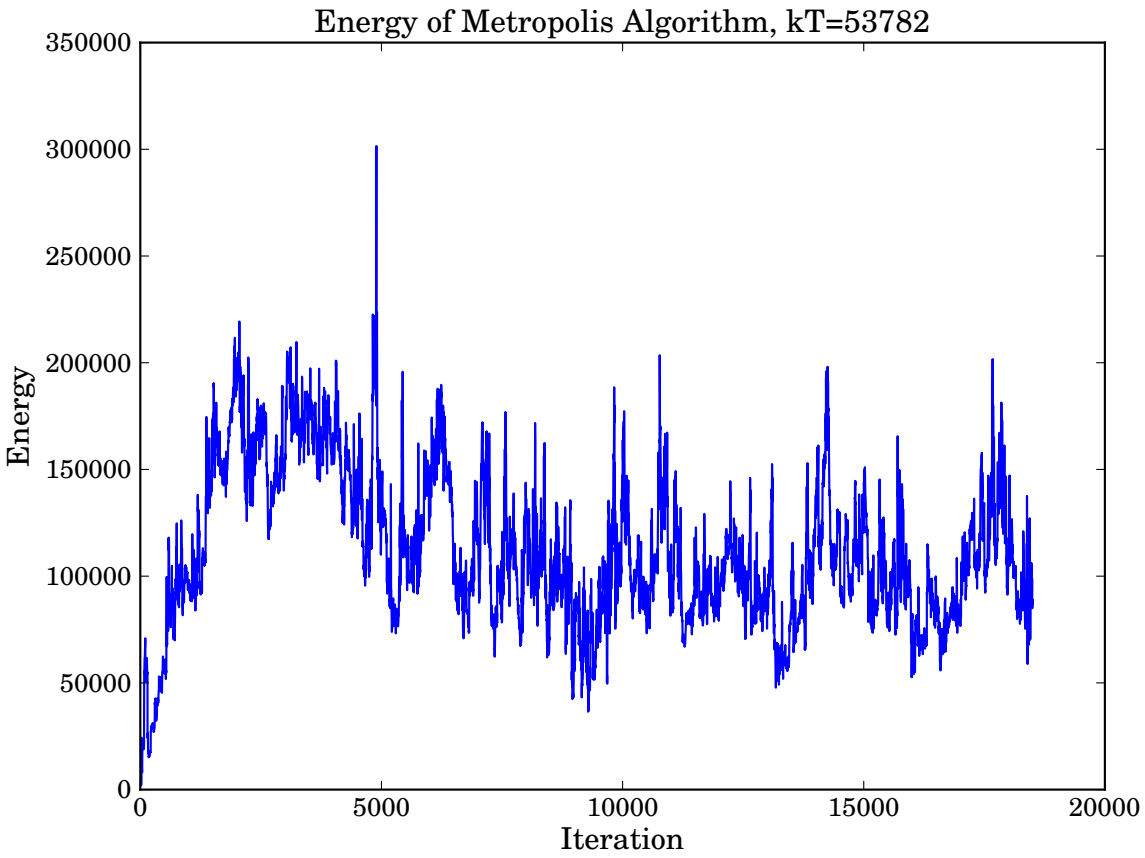
(B – 1 point) Now we'll refine `1EK8.rotated.pdb` through a Simulated Annealing approach. For the initial iterations, we aim for an acceptance criterion of 50% for structures that have twice the starting energy of `1EK8.rotated.pdb`. What should we initially set *kT* (henceforth simply referred to as the temperature since *k*, the Boltzmann constant, remains the same) to be to achieve this acceptance rate?

The energy of `1EK8.rotated.pdb` is initially $E_i = 37{,}278.5$, so an acceptance rate of 50% corresponds to:

$0.5 = e^{-(2E_i - E_i)/kT} \Rightarrow -\ln(0.5) = E_i/kT \Rightarrow kT = -E_i/\ln(0.5) \approx 53{,}782.$

(C – 2 points) Now let's implement the core of the Simulating Annealing algorithm: making a number of possible changes to the structure and accepting each according to the Metropolis criterion. To do this, fill in `onesetofMetropolismoves()`, a function that is called by `part_c()`; your implementation of `onesetofMetropolismoves()` should repeatedly call `make_backbone_change_metropolis()`, which you should implement to make one phi or psi angle perturbation drawn from a Normal(0, 20) distribution and accept or reject it according to the Metropolis criterion (see the code for more specific details).

Finally, change `kT_from_part_B = 1` at the bottom of the code to your answer from part (B). If your implementation of the functions called by `part_c()` is correct, `energy_plot_metropolis.pdf` will be produced, showing the energy of the structure at each of the `100*number_of_residues` iterations with your `kT_from_part_B`. Upload it to the Stellar dropbox or include a printout of it with your writeup.



(D – 2 points) Finally, let's implement an annealing schedule to make the Simulated Annealing algorithm more realistic:

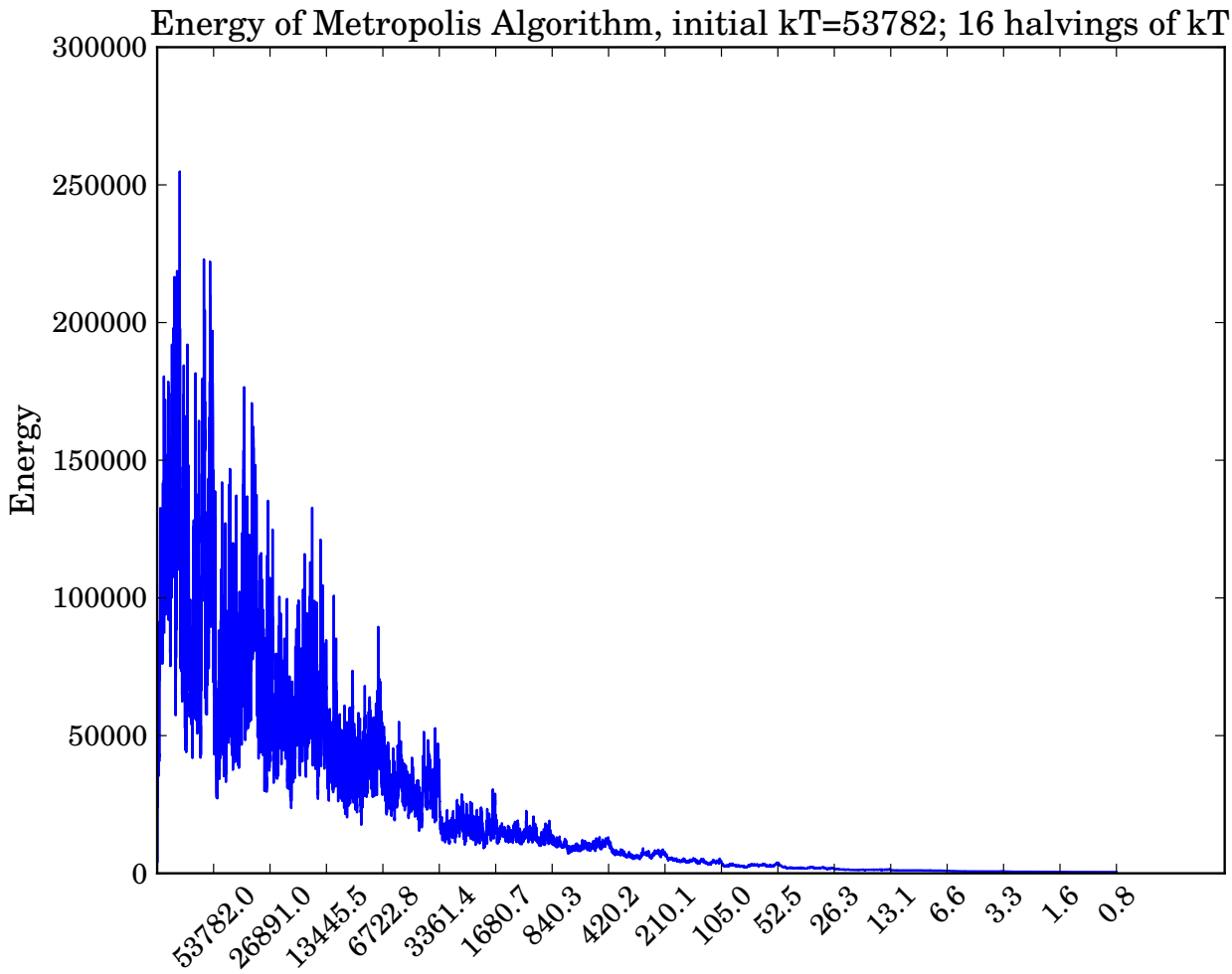
1. Start at the temperature calculated in part (B).
2. Perform `100*number_of_residues` iterations with this kT.
3. Cut kT in half. Repeat step 2.
4. Continue halving until kT is less than 1. You should perform `100*number_of_residues` iterations for the first kT value less than 1, and then stop.

Using the temperature calculated in part (B), how many kT halvings should you perform?

16 $(53{,}782/2^{16} \approx 0.82)$

In the code, change `number_of_kT_halvings = 1` to this value. Then complete the `part_d()` function to implement the above annealing schedule (see the code for more specific details). If your implementation is correct, `energy_plot_metropolis_part_d.pdf` will be produced, showing the energy of the structure at each of the `100*number_of_residues*(number_of_kT_halvings+1)` iterations with the kT at the end of each iteration set labeled on the x-axis. Upload this plot to the Stellar dropbox or include a printout of it with your writeup. In no more than two sentences, compare/contrast this Simulated Annealing plot with the Energy Minimization plot from part (A). (Note: performing the algorithm with the computed `number_of_kT_halvings` could take up to an hour, so it's recommended that you test your code with

`number_of_kT_halvings=1` first to make sure there are no errors, and then change `number_of_kT_halvings` to your calculated value to run the Simulated Annealing to completion).



The Energy Minimization algorithm from part (A) is deterministic, constantly lowering the energy and requiring fewer iterations since a change in the structure is always performed at each iteration. The Simulated Annealing algorithm from part (D) is stochastic and sometimes moves to higher energy states, requiring more iterations since many conformations are sampled but only a small percentage are accepted by the Metropolis criterion.

(E – 1 point) Finally, let's evaluate the structures you've produced using two criteria:

(1) a simple count of the number of phi and psi angles that differ by more than 1 degree.

(2) the root-mean-square deviation (RMSD).

To calculate (1), you should complete the `part_e_num_angles_different()` function. A PyRosetta function can be called to easily calculate (2) (hint: see the Appendix of the linked PyRosetta Documentation).

Then fill out the table below. Which approach (Energy Minimization or Simulated Annealing) appears to have done better here? Can you think of a scenario in which the other approach would do better?

Difference between `1EK8.clean.pdb` and:		
	Number of phi and psi angles that differ by >1°	RMSD
`1EK8.rotated.pdb`	4	5.39
Energy Minimization structure from part (A)	22	3.04
Simulated Annealing structure from part (D)	361 (will vary since Simulated Annealing is stochastic)	36.23 (will vary)

It appears that Energy Minimization resulted in a structure that is closer to `1EK8.clean.pdb` as measured by both the number of angles that differ by >1° as well as the RMSD. Since only 4 phi/psi angles were rotated in the `1EK8.rotated.pdb` structure (and these were only off by 10° each), it's not surprising that the deterministic energy minimization approach did well because the nearest local minimum found by Energy Minimization is similar to the original `1EK8.clean.pdb` structure. In contrast, the Simulated Annealing approach perturbed almost all angles during random sampling, even most of the angles which were not initially different between `1EK8.rotated.pdb` and `1EK8.clean.pdb`.

The Simulated Annealing approach would perform better than Energy Minimization when there is a large conformational change with an energy barrier between the starting structure and the globally lowest energy structure. In Energy Minimization, this barrier could not be surmounted since only downward moves on the energy surface are made and the algorithm would return a locally optimal structure that didn't make the large conformational change to get to the globally lowest energy structure. In contrast, it's possible that the random sampling in Simulated Annealing could sample something close to the globally lowest energy structure and/or temporarily move to structures with higher energy on the path to overcome the barrier and eventually settle in the globally lowest energy structure.

P3. Mutual information of protein residues (7 points).

In this problem, you will explore the mutual information of amino acid residues in a Multiple Sequence Alignment (MSA) of the Cys/Met metabolism PLP-dependent enzyme family (http://pfam.sanger.ac.uk/family/PF01053#tabview=tab0). Skeleton code is provided in `mutual_info.py` in `Problem3.zip` on Stellar. The full MSA, which you should use for all of your answers, is `cys_met.fasta`. However, we have also provided you with a smaller file containing the first ~1000 lines of the MSA (which you can use when developing and testing your code to cut down on the run-time) as `cys_met_shortened.fasta`.

(A – 2 points) Open `mutual_info.py` and scan to the bottom (after `if __name__ == "__main__":`) to get a feeling for what functions are executed and what each function returns.
First, we'll calculate the information content at each position of the alignment. What is maximum information possible at one position (in bits), and what is the formula for information that you will implement?

Maximum information possible: 4.32 bits. There are 20 possible states (amino acids) – the maximum Shannon entropy (which is also the maximum information possible - realized if only one state occurs with probability 1 and the other 19 with probability 0) is $\log_2(20) \approx 4.32$ bits.

Formula for information at a position: $4.32 + \sum_{aa=1}^{20} p_{aa} \log_2 p_{aa}$, where p_{aa} is the probability of amino acid aa.

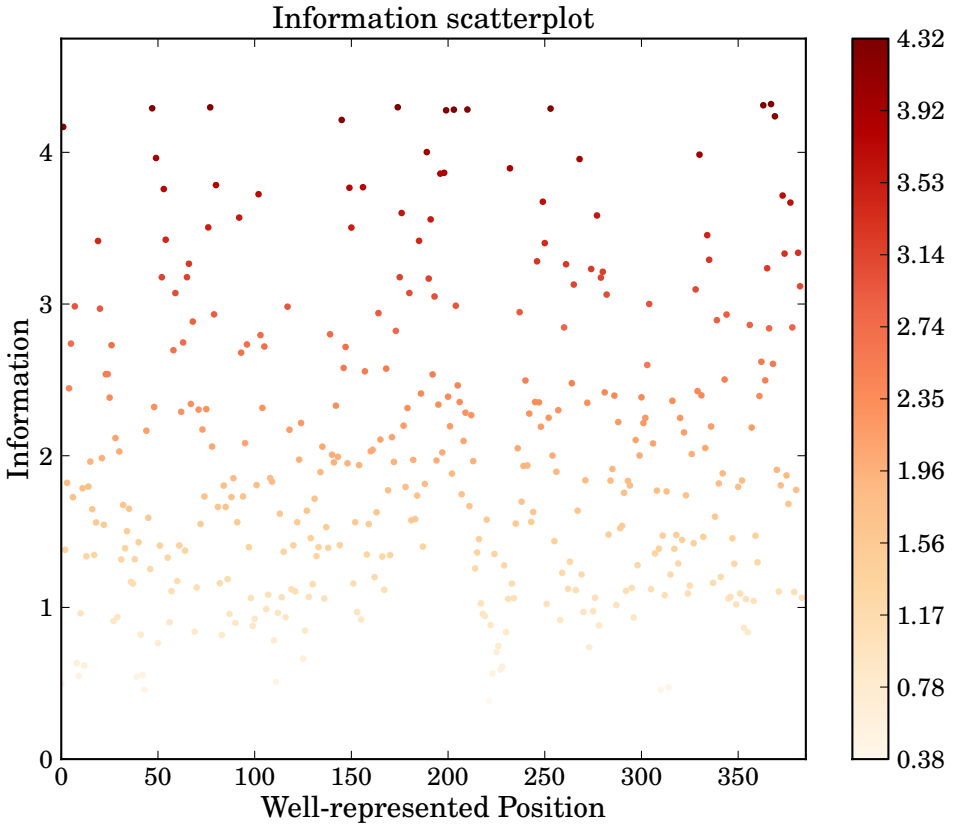
Now, complete the `part_A_get_information_content()` function in `mutual_info.py`, and plot the information content at each position (see the code for more specific details). The code can be run with the following command:

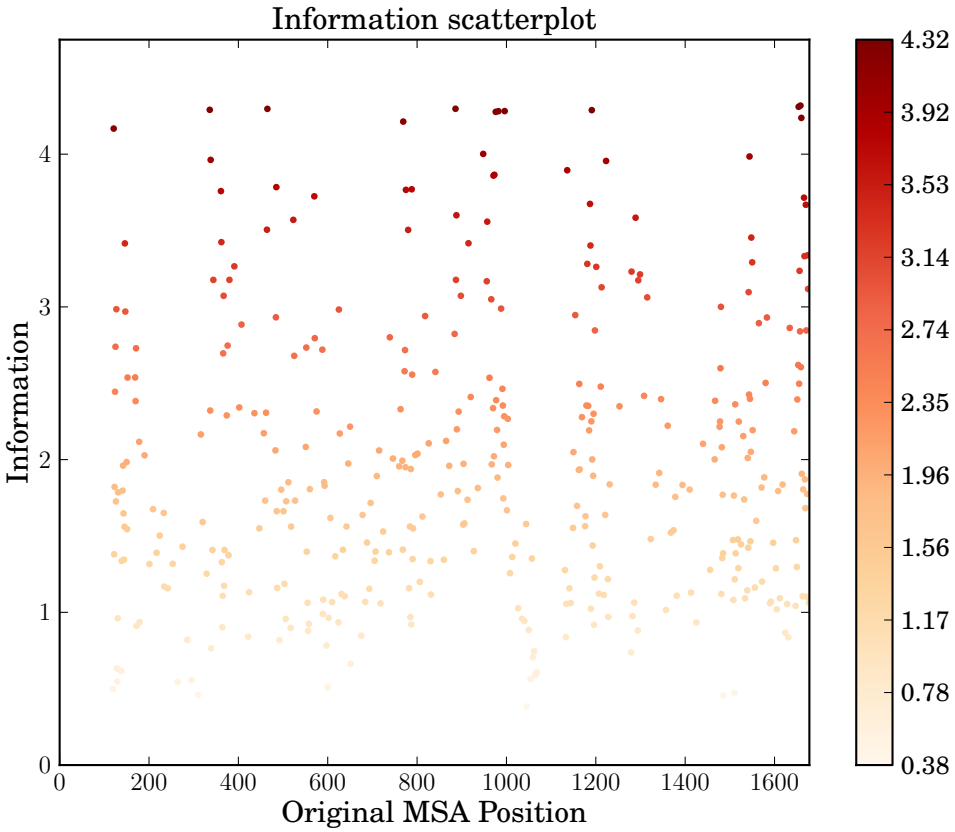
```
python mutual_info.py cys_met.fasta
```

What is the maximum information content, and what is the first position at which this maximum information is attained?

The maximum information content at any position is 4.318 bits, and it is attained at position 1658 (well-represented position 367).

If you have matplotlib installed (you also can upload your code to Athena, which has matplotlib installed), you can uncomment `plot_info_content_list(info_content_list)`, which will make 2 plots of the information content at each position (one relative to the original MSA positions and a condensed version relative to the well-represented position numbers); otherwise, make a plot of the information content at each position with a tool of your choice. Upload one of the 2 plots (or your own custom one) to the Stellar electronic dropbox or include a printout of it with your write-up.


Information scatterplot
Information
Well-represented Position
0
1
2
3
4
0
50
100
150
200
250
300
350
4.32
3.92
3.53
3.14
2.74
2.35
1.96
1.56
1.17
0.78
0.38

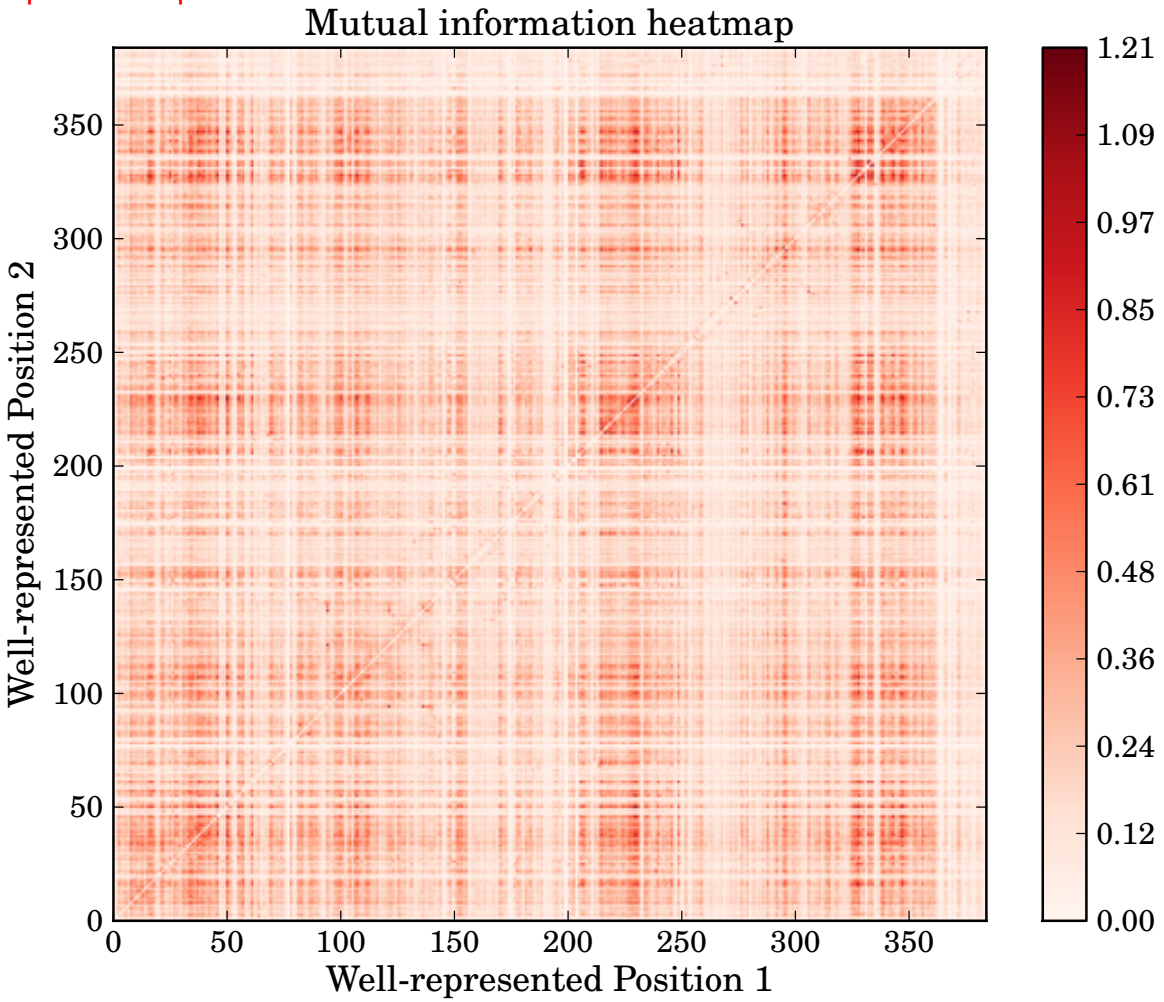

Information scatterplot
Information
Original MSA Position
0
1
2
3
4
0
200
400
600
800
1000
1200
1400
1600
4.32
3.92
3.53
3.14
2.74
2.35
1.96
1.56
1.17
0.78
0.38

(B – 3 points) Now let's calculate the mutual information between all pairs of well-represented positions in the alignment. Complete the function `get_MI_at_pairs_of_positions()` to get the mutual information at each pair of positions, and plot the mutual information at each pair of positions (if you have matplotlib installed, you can uncomment the `plot_mutual_information_dict(mutual_information_dict)` function provided to make 2 heatmap plots, one relative to the original MSA positions, and a condensed one relative to the "well-represented" position number).

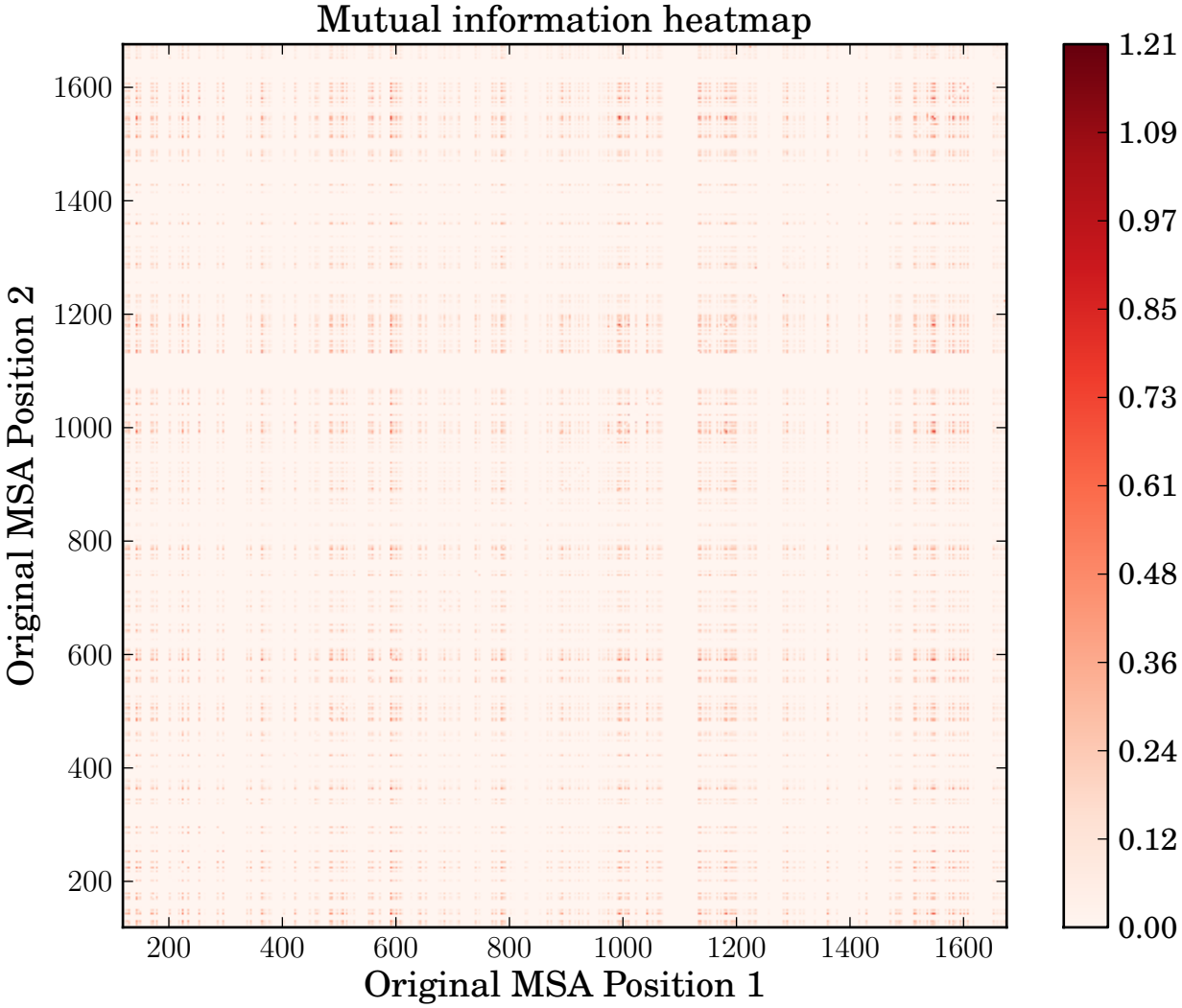
What is the maximum mutual information, and at what pair of positions is this value achieved? Upload the heatmap plot with the well-represented positions to the Stellar Problem Set 4 electronic dropbox or include a printout of it with your write-up.

The maximum mutual information is 1.212, and it is attained between positions (339, 1131) of the original MSA (well-represented position #s (50, 229)).

The condensed heatmap of mutual information for position pairs indexed relative to the "well-represented positions":



The heatmap of mutual information for position pairs indexed relative to positions in the original multiple sequence alignment:



(C – 2 points) Now's lets see if we can make sense of why the positions with the highest mutual information are as such. Complete the function `part_c_get_highest_MI_block_of_10()` to find the 10 consecutive well-represented positions with the highest average mutual information. If your function is implemented successfully, the code will subsequently print out the human sequence (including gaps and intervening non-well-represented positions) corresponding to this block. What is it?

N---R--L-R--F--L--Q----------N-SL

The human entry (`CGL_HUMAN/19-395`) in the multiple sequence alignment `cys_met.fasta` corresponds to 1QGN (Figures 1-5) in the paper "New methods to measure residues coevolution in proteins" by Gao et al. BMC Bioinformatics 2011, **12**:206 (http://www.biomedcentral.com/content/pdf/1471-2105-12-206.pdf). By matching the printed-out, gapped sequence in the MSA human entry to the ungapped human sequence (Isoform 1 of http://www.uniprot.org/uniprot/P32929), determine which positions of the human protein correspond to the highest MI block. Are these positions in any of the figures from the paper? Based on the structure of the enzyme, why would you expect these positions to have high mutual information?

As a reminder, be sure that your final answers are for `cys_met.fasta` (not `cys_met_shortened.fasta`). Upload your completed `mutual_info.py` script to the Stellar electronic dropbox.

By matching the "NRLRFLQNSL" residues with the Uniprot ungapped human sequence, we see that these are residues 234-243. In particular, the first "L" corresponds to human protein residue 236, which is the red residue in Figure 4 of the Gao et al. paper. The structure in that figure shows that residue 236 is close proximity to many other residues in the tertiary structure. Thus, if residue 236 mutates, we expect compensatory changes in the nearby residues to accommodate the new geometry of the site, explaining why residue 236 covaries and has high mutual information with others nearby in the protein.