From BMI 214

Site: Project4

On this page... (hide)

NOTES:

1. Project 4 is due electronically by **11:59PM, Thursday, December 5, 2013**.
2. We will **NOT ACCEPT ANY** projects turned in after **11:59PM, Tuesday, December 10, 2013**.
3. Please review the "style / readability" sub-section of the code expectations in the **Code Policy**.
4. Please review the **Partner Policy**. Programming projects should be completed individually. You may discuss algorithms with others, but the coding should be done alone. Any collaboration must be explicitly mentioned in header comments in your code.
5. Visit the Piazza page with any questions or problems.
6. Integrated Design: This project is significantly less challenging than projects 1-3, so try to put some extra effort into good integrated design. Specifically, read the whole of section 6 before you start coding. The three tasks require many similar or identical functionalities. One way to achieve this is to copy/paste class and function definitions from one program to the next. For the purposes of this class, this is acceptable, but suboptimal (Note: copy/pasting blocks of code *within* a program is never acceptable). To earn the final 3 points for a perfect score, you should import functions and classes from the scripts you've already written rather than copy/pasting between programs. See, for example, the section on Modules in the Python Tutorial.

1. Files to download

The 9 files needed for this assignment are contained within 3 subdirectories zipped into this file:

http://bmi214.stanford.edu/files/p4/p4.downloaded.files.zip

2. Introduction

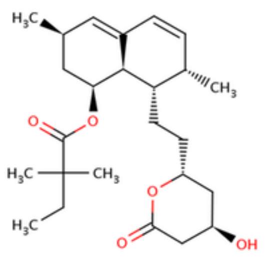
Throughout much of this course we have looked at biological data to better understand the functions of proteins and their relationships to other biological phenotypes. In this project we will be looking at chemicals and chemical structure to gain insight into how chemical similarity of protein ligands relates to the similarity of the proteins themselves.

In some cases two proteins may be highly similar in both sequence and structure, and as a result they will bind at least some of the same ligands (e.g. chemicals or small messenger molecules). However, it is also possible that two receptors may appear to be extremely different by all bioinformatic measures and yet still bind the same ligands. Similarly, there exist highly similar receptors which do not bind any of the same ligands.

So rather than look at sequence or even structure to compare proteins we can look at the sets of ligands (chemicals) which bind their receptors. This approach was used by Keiser et al in their paper **Relating protein pharmacology by ligand chemistry**. In this project we will implement a similar method to compare proteins and their ligands, look at a few specific cases of protein similarity, and then build a network connecting the proteins by the similarity of their ligand sets.

3. Drugs, SMILES, Fingerprints, and Tanimoto comparisons

The **DrugBank** database contains more than 4,000 records of small molecules, peptides, and approved and experimental drugs. Each drug in the database has a "DrugCard" which contains detailed information about the drugs, including mechanism of action, structure, function, and chemical representation. For example, the entry for Simvastatin includes an image of the structure:



It also includes a SMILES (Simplified Molecular Input Line Entry System) code for the drug:

```
CCC(C)(C)C(=O)OC1CC(C)C=C2C=CC(C)C(CCC3CC(O)CC(=O)O3)C12
```

These SMILES codes are ASCII simplifications of the drugs which provide a 2D representation of the atoms and the bonds in the molecule. You can read more about SMILES at the **Daylight Chemical Information Systems** website.

One of the advantages of these 2D ASCII representations is that they are very efficient for computation. Using the SMILES code we can compute a 2D fingerprint, which is a further simplification of the molecule into a set of *keys* indicating the presence of a feature in the molecule. To compute a fingerprint, chemoinformaticians have chosen different features of molecules that are represented by these keys. Examples of keys are:

- Are there fewer than 3 oxygens?
- Is there a S-S bond?
- Is there a ring of size 4?
- Is at least one F, Cl, Br, or I present?

We can then think about a fingerprint as a bitmap or a set where each value indicates whether or not the molecule contains that key. For example, the fingerprint for Simvastatin is:

```
26 50 57 66 74 76 83 89 91 96 99 104 108 112 114 115 116 123 126 127 128 129 132 136 137 139 140 141 143 146 147 149 150 152 154 155
```

So Simvastatin has features 26, 50, 57, ... and none of the features not listed above. You can read more about fingerprints at the **Daylight Chemical Information Systems** website.

In this project we have provided you with a list of drugs and a list of targets for those drugs. Each drug is also listed with its fingerprint. In order to compare two chemicals we can use the Tanimoto Coefficient (also known as the Jaccard Index). For two molecules mol_A and mol_B we define the Tanimoto coefficient, T_c as:

$$T_c(mol_A, mol_B) = \frac{|fpt(mol_A) \cap fpt(mol_B)|}{|fpt(mol_A) \cup fpt(mol_B)|}$$

Where $fpt(mol_A)$ and $fpt(mol_B)$ are the fingerprint sets for the two molecules A and B.

4. Ligand sets and similarity

We have already learned about using sequence alignments and scores (such as project 1 or BLAST) to compare two proteins. In this project we will be comparing two proteins by comparing the sets of ligands that bind their receptors. Using this approach we could compare two receptors A and B by comparing the pairwise chemical structure of the ligands that they bind. So if we have two molecules A and B and we know what ligands bind to them we can compute the T_c for each pair and look at the values. The **Similarity Ensemble Approach** (SEA) of Keiser et al computed a stastic (similar to BLAST) in order to determine the significance of these scores. For this project, we will use a simplified version.

Their statistic involves computing a similarity score by summing up all of the pairwise T_c values above a cutoff. If the T_c value is too low, then it mostly contributes noise to the data. For our project we will define the cutoff as 0.5.

So, we define the $T_{summary}$ for two proteins A and B as:

$$T_{summary}(A, B) = \sum_{\substack{a \in targets(A) \\ b \in targets(B) \\ T_c(a,b) > 0.5}} T_c(a, b)$$

In other words, compute all the pairwise Tanimoto similarities for the drugs that bind protein A and protein B, and sum the values that are greater than 0.5.

So if A binds ligands a_1, a_2, a_3 and B binds ligands b_1 and b_2 then we have to compare:

- a_1 -- b_1
- a_1 -- b_2
- a_2 -- b_1
- a_2 -- b_2
- a_3 -- b_1
- a_3 -- b_2

We then sum up all of the T_c values that are greater than 0.5.

This number will depend upon the size of the ligand sets, so it can be difficult to compare these values for significance. We will generate a **bootstrap p-value** in order to determine which $T_{summary}$ values are significant. We do this by choosing 100 random pairs of sets of ligands. One set should be the same size as the ligand set for molecule A and the other should be the same size as the ligand set for B. We then compute $T_{summary}$ for each of them. This will give us a sense of how likely it would be to get our value of $T_{summary}$ at random. If there are 5 random set pair comparisons that have a value of $T_{summary}$ that is greater than or equal to our real (non-randomized) calculation, then the bootstrap p-value is 5 / 100 or 0.05. We can interpret this to mean that, under the null hypothesis, there is a 5% chance that we would see this value of $T_{summary}$ or greater.

In general, for N iterations we calculate the $p_{bootstrap}$ with:

$$p_{bootstrap} = \frac{\#\ of\ T_{random} \geq T_{summary}}{N}$$

where N = 100 in the above example. So if we are given given two proteins A and B and we want to determine if their ligand sets are significantly similar we will:

1. compute $T_{summary}(A,B)$ as above for the proteins using their real target sets
2. let na = size(targets(A)), nb = size(targets(B))
3. repeat N times
   - choose A_{random} to be a set of na random ligands
   - choose B_{random} to be a set of nb random ligands
   - compute $T_{summary}$ using these random ligand sets
4. compute $p_{bootstrap}$ using the equation above

5. Generating and Visualizing Networks

5.1 Description

In the last part of this project you will generate bootstrap p-values for a subset of drug target proteins. Look at the file `protein_nodes.csv`. For a subset of swissprot IDs from `targets.csv`, it lists the usual indications for which the protein is targeted (cholesterol, bp, and/or diabetes).

Using the bootstrap p-values that you compute using the above method, you will create a network where the nodes are proteins and edges are drawn between pair for which the bootstrap p-value is less than or equal to 0.05. You will do this by generating a network file and loading it into Cytoscape, a popular network visualization tool.

You should compute your bootstrap p-values using the entire set of drugs in drugs.csv for your random sets. However, you will only be generating the network nodes for the proteins in `protein_nodes.csv`.

You will generate three files for Cytoscape: a `SIF` file and two `nodeAttr` files.
1. The `SIF` file links pairs of SWISSPROT IDs with edges. It look like this:

```
P02768 edge P27169
Q9Y478 edge Q13131
Q9Y478 edge O95477
P33993 edge P04114
P33993 edge Q92887
...
```

Format requirement: This file should be sorted within AND between rows. That is, the protein ID on the left should be alphabetically less than the ID on the right, and column 1 should be a sorted list. Points will be deducted otherwise.

2. The two `nodeAttr` files provide labels for the nodes.
`name.nodeAttr` labels the nodes with the SWISSPROT name from `protein_nodes.csv` like this:

```
name
P02768 = ALBU_HUMAN
P27169 = PON1_HUMAN
Q9Y478 = AAKB1_HUMAN
Q13131 = AAPK1_HUMAN
O95477 = ABCA1_HUMAN
P33993 = MCM7_HUMAN
P04114 = APOB_HUMAN
Q92887 = MRP2_HUMAN
...
```

3. `indication.nodeAttr` labels the protein with a semi-colon separated list of all the indications associated with a given protein (also in `protein_nodes.csv`). This file should look like this:

```
indication
P02768 = bp;diabetes
P27169 = bp;cholesterol
Q9Y478 = bp;diabetes
Q13131 = bp;diabetes
O95477 = bp;diabetes
P33993 = bp;cholesterol
P04114 = bp;cholesterol
Q92887 = bp;cholesterol
...
```

Note that there can be many indications associated with a drug and its targets, but for this project we will just look at one primary indication per node.

The final file for Cytoscape you will use is `p4.vizmap.props` file, which defines a style for the network such as how to label and color the nodes. We have provided this file for you.

5.2 Cytoscape Example

Download and install **Cytoscape** now. Have our example files on hand:

- example_network.sif
- example_indication.nodeAttr
- example_name.nodeAttr
- p4.vizmap.props

To test our example network:

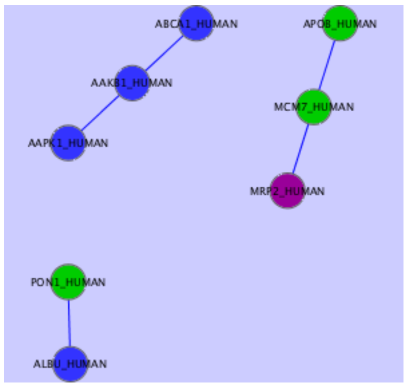
1. Launch Cytoscape and choose "File:Import:Network (Multiple File Types) ..."
2. Select "example_network.sif"
3. Select "Import" and then "Close"
4. Choose "File:Import:Node Attributes..."

5. Select "example_indication.nodeAttr"
6. Choose "File:Import:Node Attributes..."
7. Select "example_name.nodeAttr"
8. Choose "File:Import:Vizmap Property File..."
9. Select "p4.vizmap.props" and then "Close"

Now you can choose a layout. For this project we will use "Spring Embedded". To get this:

- Choose "Layout:Cytoscape Layouts:Spring Embedded".

You should see something like this:



There will be some variability in the layouts. The Vizmap Properties that we have set will:

- Color Blood Pressure indication as red.
- Color Blood Pressure + Cholesterol indication as green.
- Color Blood Pressure + Diabetes indication as blue.
- Color Blood Pressure + Cholesterol + Diabetes indication as purple.
- Label the nodes with the SWISSPROT Name.

6. Project Programs

6.1 Tanimoto Generation

Write a program named `tanimoto.py` that reads in `drugs.csv` and generates the pairwise Tanimoto similarity scores for all pairs of drugs. Indicate if the drug pair are known to share a target, as listed in `targets.csv` file.

Your program should take 3 command line arguments:

```
tanimoto.py drugs.csv targets.csv outputfile.csv
```

The output CSV file should look like:

```
DB00001,DB00006,0.810127,1
DB00001,DB00007,0.835443,0
DB00001,DB00010,0.691358,0
...
```

The first two columns are the DrugBank IDs, the third is the Tanimoto score (round values and print exactly six decimal places), and the fourth/last is 1 if the pair of drugs share a target and 0 if they do not. Each pair of drugs should only occur once. So include (a, b) but not (b, a). Also, the list should be sorted by col1 and then by col2.

Run your program, and then generate 3 histograms, titling each as indicated:

- `all_tanimoto.png` all Tanimoto values
  Title = "{sunetID} All"
- `shared_tanimoto.png` Tanimotos for ligands that share a target
  Title = "{sunetID} Shared"
- `notshared_tanimoto.png` Tanimotos for ligands that do not share a target
  Title = "{sunetID} Not Shared"

6.2 Ligand Set Similarity

Write a program called `pvalue.py`. It will generate a bootstrap p-value for the comparison of two proteins.

Your program should read `drugs.csv` and `targets.csv` to get all of the fingerprints and protein targets for each drug. It should take one input option and four arguments.

- option: -n <int> for the number of iterations
- arguments 1 and 2: the `drugs.csv` and `targets.csv` file paths
- arguments 3 and 4: two SWISSPROT IDs

```
pvalue.py -n <INT> <drugs.csv> <targets.csv> <proteinA> <proteinB>
```

By default n = 100, so with no option the program will display the bootstrap p-value from 100 iterations:

```
$ pvalue.py drugs.csv targets.csv P21918 P18089
0.17
```

With n = 1000, the program will display the bootstrap p-value from 1000 iterations:

```
$ pvalue.py -n 1000 drugs.csv targets.csv P21918 P18089
0.205
```

Note that there will be some variability in p-values due to the random selection of ligand sets.

6.3 Network Visualization

Write a program called `networkgen.py` that generates:

- `network.sif`
- `name.nodeAttr`
- `indication.nodeAttr`

for all pairs of proteins in `protein_nodes.csv` that have a bootstrap p-value ≤ 0.05. You should use all of the data in `drugs.csv` and `targets.csv` to generate your bootstrap p-values, but only use the protein nodes listed in `protein_nodes.csv` for the pairwise computations.

Your program should take 3 arguments for the path to `drugs.csv`, `targets.csv`, and `protein_nodes.csv`.

```
 networkgen.py drugs.csv targets.csv protein_nodes.csv
```

Do NOT compute links from proteins to themselves. Also, note that Tanimoto coefficient is symmetric, so you only have to compute one direction, and you should only write edges in one direction. For example if you have:

```
P02768 edge P27169
```

DO NOT also write the other direction:

```
P27169 edge P02768
```

This computation will probably take a few minutes. Be sure to only include nodes and edges with significant p-values (≤ 0.05).

Load the resulting files into Cytoscape along with `p4.vizmap.props`. Use the "Spring Embedded" layout to visualize the network. Export the image as a PNG using "File:Export:Network View as Graphics ...".

6.4 Project 4 Quiz

Use the template `project4_quiz.txt` to answer all quiz questions `project4_quiz.YEAR.pdf`.
Answers must be no more than two sentences per question. Too long = wrong.

7. Deliverables

Your submission will include 12 or 13 files:

- 1 [optional] utilities file containing your class and function definitions:
  `chemoUtils.py`
- 3 source code files:
  `tanimoto.py`, `pvalue.py`, `networkgen.py`
- 3 histograms of Tanimotos:
  `all_tanimoto.png`, `shared_tanimoto.png`, `notshared_tanimoto.png`
- 3 cytoscape files:
  `network.sif`, `name.nodeAttr`, `indication.nodeAttr`
- 1 cytoscape network image file:
  `network.png`
- 1 `README.txt` This should be a text file that includes these sections.
  - Name and SUNet ID
  - Language used and instructions on how to run your program
- 1 `project4_quiz.txt`

8. Submission instructions

Warning

Points will be deducted for infractions of ANY of these guidelines:

- File names should be EXACTLY as indicated above.
- Submit ONLY the 12 (or 13) files listed above.
  Remove all other files before submitting, including `*.pyc`.
- All files should be contained within the directory from which you call the submission script.
  NO subdirectories.

Instructions

1. Transfer all 12 (or 13) files into one directory on corn.
   For example: `~/biomedin214/p4/toSubmit`
2. Log in to corn: `ssh sunetID@corn.stanford.edu`
3. cd into your submission folder: `cd ~/biomedin214/p4/toSubmit`
4. Test your program to ensure that it works on corn.
5. Remove every file other than the 12 (or 13) to be submitted. Don't forget to `rm *.pyc` and remove all data files.

6. Run the submission script:
   `/usr/class/biomedin214/bin/submit p4`
   Note the space before `p4`.
   Be sure to run the script from within your submission folder (the folder containing the deliverables)!
   You may submit multiple times; simply re-run the script. Each new submission overwrites the previous submission. Your submission date will be the final submission received, and late hours will be charged accordingly.