Biological Data Analysis (CSE 182) Final projects

1 Logistics

The final 1 or 2 lectures of the class will be devoted to final presentation of the project. You are required to work on one of the projects described below. Checkpoints have been created to help you stay on track. They will not be graded separately, but you must meet the deadlines. Soon after your first checkpoint, you should schedule a meeting with the instructor, preferably at the discussion time, Monday 11/17, or Tuesday 11/18. Students choosing 1.3 must also submit A4. Otherwise A4 is optional.

Checkpoints

C1:11/10/08: Submit a 1-2 page written report with answers to the following questions:

1. Your project partner’s name, if any. Teams of 2 people are recommended, but 1 or 3 will also be accepted.
2. Your choice of the project. Choose from one of the following, or from ubergrid.org.
3. For each project, the ability to read input files, available after 11/11.

C2:11/17/08: Answers to the C2 part of your project. Schedule a meeting with the instructor to discuss final project.

C3:12/1,3: C3 part, and final presentation.
1.1 Homology based gene finder

**Goal:** Given a protein sequence from a species, find the location of the orthologous gene using genomic data from another species (Ideal group size: 3 people)

**Skills:** Scripting/programming. The tool could be built entirely using public domain tools such as Blast, and a scripting language.

**Motivation:** As more genomes are sequenced, initial gene identification is based on such comparative analysis. We want to look at the current maize genome, using protein sequences from other plants, like rice.

**Questions:** Answer the following:

1. Download the maize genome from XXX. How large is the available data-set? How many sequences (C1)?
2. Download the rice proteome from XXX. How large is the data-set? How many sequences?
3. Blast each rice protein against the maize genome. What version of Blast will you use? How many queries get at least one hit (C1)?
4. For each rice sequence, decide if ortholog is completely present in the genomic database, partially present, or absent. Develop code that takes the Blast hits, and builds the following OrthologTable: each row has 4 columns: the protein ID for rice, an identifier for the genomic sequence, strand (+ve/-ve), and an indicator set to 0 (no-hit), 1 (partial-hit), or 2 (over 90% of the sequence is hit) (C2).

5. Build a gene finder as follows:

   (a) For each protein query, use the Blast output to identify a genomic region that contains the ortholog. Remember that the query might hit multiple locations (paralogs), and genomic sequence from multiple contigs. Separate the different paralogs, and order the contigs within each paralog (C3).

   (b) While Blast should help locate most of the exons, it might miss some of the smaller exons, and get the boundaries incorrectly. Use the tool Augustus to reconstruct the best transcript (C3).

   (c) Build a table of exons coordinates. Each row contains the following: Rice protein Id, genomic ID, strand, begin-exon coordinate, end-exon coordinate (C3).

   (d) Concatenate and translate the exons to get a protein sequence. Blast the rice query against the translated amino-acid sequence to check similarity (C3).
6. **Redo OrthologTable.** For each rice sequence, decide if ortholog is completely present in the genomic database, partially present, or absent. Develop code that takes the Blast hits, and builds the following table: each row has 4 columns: the protein ID for rice, an identifier for the genomic sequence, strand (+ve/-ve), and an indicator set to 0 (no-hit), 1 (partial-hit), or 2 (over 90% of the sequence is hit) (C3).

**Presentation:**

1. Describe the project.
2. Provide general statistics (answers to the questions above).
3. Show interesting cases where the gene finder could identify the gene, or failed to reconstruct the final gene.
4. Future work.
1.2 Post-translational protein analysis

**Goal:** The goal of this project is to identify post-translational modifications and protein expression levels that are condition/tissue specific.

**Skills:** The programming skills needed here are minimal, but require you to use web-resources or freely available tools such as R. This is a more open ended project, and points will be given for creativity.

**Motivation:** A comprehensive search of the Arabidopsis proteome resulted in identification of a number of proteins, and their modifications. The data were acquired from different tissues. Additionally, peptides were identified from root with and without infection from a nematode. It is expected that different proteins are expressed preferentially in different tissues/conditions. The goal of this project is to help visualize this massive data-resource. As a simple example, compare different tissues/conditions by making a heatmap of protein expression.

**Questions:**

1. Collect peptide/protein data from different Arabidopsis tissues: leaf, silique, root, etc. by emailing Natalie Castellana ncastellana@ucsd.edu (C1).
2. Identify a web resource, or learn how to draw a heat-map using the R programming language or Python (C1).
3. Build heat maps and other visualizations using spectral counts of expressed proteins, and their post-translational modifications. If you select with the project, meet with the instructor in the week of 11/9 to discuss.
1.3 Dictionary matching tool

Students choosing this project must also submit A4.

**Goal:** Build a robust dictionary matching tool.

**Skills:** Good programming skills (Java/C++)

**Motivation:** Dictionary matching is an important part of a string matching toolkit. In the class, we learned only to search with an existing dictionary. Here, we will build a tool for constructing a dictionary, and searching with it.

**Questions:** Answer the following (C1):

1. What is a trie?
2. What data structure will you use to encode the trie?
3. What data structure will you use to encode the failure function?
4. Describe algorithms to search with a trie, and to build one.
5. Implement code `searchtrie` to search a multi-fasta file with a trie. If a sub-sequence from sequence $S$ matches the dictionary, you must report the header of the sequence, and the position in the sequence.

**Presentation:**

1. Implement code `buildtrie` to build a trie, given a dictionary of keywords (C2).
2. Run `buildtrie` and show your output for various dictionaries provided (C2).
3. Run `searchtrie` using the provided dictionaries, and
4. Generate random database strings of size 1K, 10K, 100K, 1M, 10M, 100M, 1B letters. Embed words from the dictionary in it. Verify correctness of search by running `searchtrie`. How does the running time increase with an increase in database size, dictionary size (C2).
5. Create random dictionaries with longer and longer word-sizes $W$ ($3 \leq W \leq 50$). Search a large random string $D$ of $10B$ letters with it. Note that $D$ may not fit in main memory. Plot the number of hits as a function of $W$ (C3).
6. Generate all possible 11-mers from the input query sequence provided. Search the human genome with a trie constructed of all of those words. Does the number of hits match what you’d expect by chance? (C3)
7. The size of a dictionary is the number of (non-failure) edges in the dictionary. Plot the time taken to build the trie as a function of the the size of dictionary. To do this, generate a number of dictionaries of increasing size, by randomly generating keywords. Does your procedure grow linearly with the size of the dictionary? (C3)