Biocomputing2 Project - Database layer

Author: Igor Ruiz de los Mozos
i.mozos at ucl.ac.uk
https://github.com/jurnho/biocomp2

1. Introduction

This is the final project for the Biocomputing2 module of the Master in Bioinformatics at Birkbeck College. We start the project settling communication procedures, time line, meetings schedule, APIs, outputs and HitHub account.

2. Schema

My part is the database layer and my code is under the database folder:

```
├─── RUN_DB.sh                    # Script to install, parse and database populate
├─── database
│   ├─── doc
│   │   ├─── commentary.doc
│   │   ├─── commentary.pdf
│   │   └─── documentation.txt
│   ├─── readme.txt               # How to install the database
│   └─── src
│       ├─── create.sql                  # Create mySQL tables
│       └─── perl
│           ├─── Biocomp2
│           │   ├─── DataAccess.pm   # Subroutines to query the database
│           │   ├─── parser.pm       # Subroutines to parse the genbank data file
│           │   └─── save_DB.pm              # Subroutine to save entries to database
│           ├─── parse_genebank.pl   # Parse genbank file
│           ├─── testing.pl
│           ├─── testing_gene_detail.pl
│           ├─ ─ ─                    testing_get_aa_sequence.pl        #  Queries
output test structures
│           ├─ ─ ─                    testing_get_coordinates.pl
│           ├─── testing_get_genes.pl
│           ├─── testing_get_sequence.pl
│           └─── testing_search.pl
```

3. Code documentation

- **RUN_DB.sh**

  Script calls mysql to create the data base tables, download the genbank file, parse the file, remove the raw data and run deploy and environment settings.

- **create.sql**

  I have created two tables to contain all the data needed for this project. The tables are deleted ad created every time new prior to populate them.

  - ***Table chromosome16_genes.***

    Each record must contains gene identifier (gene_ID, primary key), accession version (acc_ver), coding strand (complement), nucleotide sequence (sequence) and codon start (cod_start). Some of the entries lack of gene name (gene), locus map (map), protein product (product) and protein identifier (protID) therefore I assign "NA" if they are empty. The

  - ***Table coordinates.***

    Every gene identifier must contain exon count number (exon_count), exon coordinate start (COO_start) and exon coordinate end (COOR_end). This table is join with the primary table using gene identifier as a foreing key.

- **Parse_genebank.pl**

  Main script for parsing file. It calls subroutines from Biocomp2::parser Biocomp2::save_DB and packages:

  - ***Biocomp2::parser::open_file***

    Open file or die.

  - ***Biocomp2::parser::get_entry***

This script uses the end of entry in a genbank file (//), to split each entry record. To get this done it reset the perl record separator ($/) that its end of line (\n) when it find genbank end of entry (\\) and re_reset the record separator again. This allow to get each entry separate in a scalar with the end of lines included.

- ***Biocomp2::parser::split_entry***

  This subroutine match on each entry everything after "LOCUS", "FEATURES" and "ORIGIN" and split it on three different scalars annotation ($annot), features ($feat) and sequence ($seq). It also remove the blanks and numbers from sequence.

- ***Biocomp2::parser::parse_annotation***

  This get annotation scalar from previous subroutine and transform to array using the start of the line. After that a for loop try to mach on each element of the array for "LOCUS", "ACCESION" and "VERSION" where it get the gene identifier and the accession version.
  #return($gene_ID, $acc_ver, $locus);

- ***Biocomp2::parser::parse_features***

  This was the most challenging part of the code. On the features region it match for CDS including new lines but / until / in a multiline approach. Then it extracts the rest of the values matching for the characters of each one getting everything after them that are quoted (example match everything quoted after /map=)
  # return (\@cordinates, $complement, $gene, $map, $cod_start, $product, $protID, $aa);

  Coordinates (@coordinates)

  To get the coordinates after removing "join" characters I have match for complement to assign to TRUE meaning that the coding regions are on the negative strand. Then I have split it based on coma to get the coordinate start and end for each exon. Since on the project its explicit that we don´t have to deal with alternative splice variants I have remove coordinates that require to join with a different accession or entry. Finally I have push the coordinates to and array of arrays. For the positive strand the approach is the same but the complement variable is assign to FALSE.

- ***Biocomp2::save.DB:: save_genes***

  Parser get all the data in two hashes and reference them to pass to save genes subroutine. This routine gets each element of the gene hash to

insert them on chromosome16_genes and coordinates hash in coordinates table on the database.

- **Biocomp2::DataAccess package**

  This package contains all the database queries:

  - ***Biocomp2::DataAccess::get_genes***

    This subroutine query all the gen entries and return a hash of arrays in where the key is the gene identifier and the values are accession version, gene name, locus map, protein product and protein identifier.

  - ***Biocomp2::DataAccess::get_gene_names***

    Query the database for all the entries and return a hash in where the key is the gene identifier and the value is the accession version.

  - ***Biocomp2::DataAccess::get_gene_details***

    Query the database for a specific gene identifier and return a hash of arrays referenced with the key as gene identifier and values as accession version, gene name, locus map, protein product and protein identifier.

  - ***Biocomp2::DataAccess::get_sequence***

    This DBI script query the database for a specific gene identifier and return the associated nucleotide sequence.

  - ***Biocomp2::DataAccess::get_aa_sequence***

    This DBI script query the database for a specific gene identifier and return the associated amino acid sequence. This was not required for the project but was implement to test that the DNA translation layer was correct.

  - ***Biocomp2::DataAccess::get_coordinates***

As before this query looks for a specific gene identifier and return a join query in the two tables. The output is pushed to an array of arrays containg the exon number, exon start, exon end, coding strand and codon start.

- ***Biocomp2::DataAccess::get_search***

  This subroutine queries the database for any characters like the search on columns protein product, gene identifier, locus map or accession version and return a hash of arrays with the key as gene identifier and the values accession version, gene name, protein product and protein identifier.

4. Commentary/reflection

This project was the most challenging coding project that I had faced. Last year I have learned Perl at class but at work they force to learn Python. At first this duality makes me quite confuse but at the end it enforce my coding knowledge and I realize that it helps me to understand how to code. During this project I have learned how to structure a complex project and how to join efforts with others mates. HitHub was a handy tool to satisfactory complete it and I'm sure I´ll use it on my day a day coding. I have also start using jetbrains IDEs to help on my coding.

The relation with the other class mates was excellent and everyone was suitable for meeting and hard work. At this point I have realize of how bioinformatics soak from the knowledge of IT and biologist. Everyone was help and assist the rest on their weakness and that encourage us to complete the project and start a friendship outside of the classroom. In really satisfied with this project and with the values and know how learned.