Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Biotecnologia
Laboratório de Bioinformática e Proteômica

GENIX: Automated Bacterial Genome Annotation Pipeline

Manual

Msc. Frederico Schmitt Kremer
Dr. Marcus Redü Eslabão
Dr. Odir Antônio Dellagostin
Dr. Luciano da Silva Pinto

2016

Genix

About

Genix is an online automated pipeline for bacterial genome annotation. The program takes a FASTA file containing a set of sequences that can be complete chromosomes, contigs or scaffolds, and a tax_id identifier. First, a dataset of proteins associated to the tax_id "Is downloaded from Uniprot and used to build a raw dataset, wich may contain several redundancies. CD-HIT (Li & Godzik 2006) is used to build a non-redundant dataset, wich is used to generate the final BLASTp (Altschul et al. 1990; Camacho et al. 2009) database. For the genome annotation, genix uses a combination of several bioinformatics tools, including Prodigal (Hyatt et al. 2010), BLASTp, tRNAscan-SE (Lowe & Eddy 1997), RNAmmer (Lagesen et al. 2007), Aragorn (Laslett 2004), HMMER (Eddy 2011), BLASTn and INFERNAL (Nawrocki et al. 2009), RFam (Griffiths-Jones et al. 2003), Antifam (Eberhardt et al. 2012) and the non-redundant dataset generated by CD-HIT. At the end, genix generates a genbank file, containing all the features identified for each sequence, and, if requested by the user, a the genbank submission file (.sqn) generated by tbl2asn.

This package contains the source code of the genix's annotation pipeline (genix_annotation.py), some scripts for database preparation and for file formatation (located in the "scripts" directory), and some of the binaries required to run Genix properly already compiled for Linux 64 bits platforms.

Additionally, we have also added a demo version of the API we are currently developing to allow batch submissions and commandline accession to some of the functionalities of the Genix server ("WebAPI" directory). This folder contains a python script that uses the third-party "requests" module to submit genomes, check the status and retrieve the results in Genbank format. Although still very limited, we believe this tool will already be useful to perform batch submissions in just a few Python lines of code.

2. Requeriments

Operating System: Linux
Python libraries: `Biopython` and `requests` (for the WebAPI script)
Software: RNAmmer, CD-HIT, makeblastdb.

3. Editing the source code

- Edit the variables "`PERL_PATH`" (path to the Perl executable), "`PYTHON_PATH`" (path to the Python executable) and "`RNAMMER_PATH`" (path to the RNAmmer directory) are correctly specified.

- Check if the binaries Prodigal ("`bin/prodigal/`"), Aragorn ("`bin/Aragorn`"), hmmer3 ("`bin/hmmer3`"), INFERNAL ("`bin/infernal/src`"), BLAST ("`bin/ncbi-blast`") and tRNAscan-SE ("`/bin/tRNAscan-SE`") are set as executable files. If not, you can use the command "`chmod +x DIR/*`" in each of the directories to provide executable permission. In some cases it may necessary to recombine some of these binaries, especially if you running then in a non-64 bits platform.

4. Running

4.1 Preparing the database

```
$bash scripts/get_database.sh <tax id> <database prefix>
<identity>
```

`<Tax id>`: Unique identifier for a given taxonomic group that is used by a wide variety of public biological databases. The tax id of a taxon can be obtained from the NCBI Taxonomy database (www.ncbi.nlm.nih.gov/taxonomy)

`<database prefix>`: The name of the BLAST database that will be generated at the end of the process.

`<identity>`: The sequence identity threshold that will be used by CD-HIT to group the protein sequence into clusters.

Example:

```
$bash scripts/get_database.sh 171 leptospira_db 0.95
```

It creates a database from all proteins with the “171” tax id (Leptospira genus) present in the Uniprot database, runs CD-HIT with a identity threshold of 95% and format the BLAST database using the prefix “leptospira_db”.

4.2 Running the annotation (simple execution)

```
python genix_annotation.py -i <contigs .fasta> \
                           -dbp <database prefix> \
                           -o <output directory>
```

The full list of available arguments:

Argument	Value
-h	Display the “help” menu
-i	Input genome (.fasta file)
-o	Output directory
-t	Number of threads to be used
-antie	e-value threshold for HMMER in the Antifam search
-blaste	e-value threshold for BLAST in the Protein BLASTs
-blastrnae	e-value threshold for BLAST in the ncRNA BLAST
-nfernale	e-value threshold for cmsearch (INFERNAL)
-gc	Genetic code table to be used by Prodigal (“11” or “4”)
-sl	The prefix of the name that will be used for each contig
-lt	Locus_tag prefix
-nst	Name of the institution (will be used in the protein ID)
-dbp	Name of the BLAST protein database to be used *

* = Use the “database prefix” passed to “get_database.sh” in the database preparation step.

5. WebAPI Example (demo)

Along with the source code we have added an python script that serves as example of how to submit a genome to the Genix server using a commandline interface. The script “GenixWebAPI.py” located in the directory “WebAPI” uses the `requests` to send and retrieve data form the server by using HTTP POST requests. We are still working on the development of the API, but it may already be useful for some tasks.

Submission

First, create a dictionary object with the following information (yes, in this version it is necessary to add all values, but we are working on it, so as soon as possible it will change).

```
submission_data = {
    'user_email' : 'fred.s.kremer@gmail.com',
    'user_password' : '1234',
    'genome_FASTA' : open('/home/cdtec/Frederico/genix/genix_annotation/'
                          'versions/0.1/webAPI/leptospira.fasta').read(),
    'locus_tag_prefix':'GNX',
    'sequencing_center':'GNX',
    'genetic_code_table':'11',
    'database_construction':{
        'tax_id':'171',
        'clustering':'95'
    },
    'organism' : {
        'organism' : 'Leptospira interrogans',
        'serovar' : '',
        'serogroup' : '',
        'serotype' : '',
        'isolate' : '',
        'isolation_source':'',
        'strain':'',
        'host':''
    },
    'evalue_thresholds':{
        'protein_blast':0.0005, # float number, usually between 0 and 1
        'rfam_blast':0.0005, # float number, usually between 0 and 1
        'antifam_hmmer':0.0005, # float number, usually between 0 and 1
        'rfam_infernal':0.0005}
    }
```

Now, send it to the server as a JSON string.

```
return_data = requests.post("http://labbioinfo.ufpel.edu.br/cgi-
bin/genix_api.py",data={'mode':'submission','data': json.dumps(
                                    submission_data)})
```

The return_data receives the value that is passed by the server after the request. In not error occurred, the `return_data` will receive a JSON string, that can be converted to a dictionary using the JSON module. The JSON object contains the JOB_ID of the submitted genome, so you may use it to retrieve the result in a future moment.

6. Contact and Feedback

In case of any problem, or if you have any idea to improve the pipeline, please, contact us!

Frederico Schmitt Kremer,
fred.s.kremer@gmail.com / fredericok.cdtec@ufpel.edu.br

Luciano da Silva Pinto,
dmpluc@ufpel.edu.br / ls_pinto@hotmail.com