Genix: An automated pipeline for bacterial genome annotation

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 repository 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.

Be aware

The *ab initio* gene prediction in microbial genomes is usually pretty accurate, but we strongly recommend manual revision for those cases where there is the intention to perform a deeper analysis of some specific gene. Usually the most common errors found in microbial genome annotations are associated with wrongly-identified start codons, as is possible to have two or more start codons (eg: ATGs) in the same frame, and located close to each other. To reduce this type of error, we have a added a start codon correction step after the protein annotation, but even after this step it is also recommended a manual curation, as the protein databases, specially those that are not manually curated, are full of misannotated proteins.

Requeriments

Operating System: Linux (eg: Ubuntu) 64-bits

Python Libraries: BioPython (sequence operations) & Requests (for the WebAPI)

Perl libraries: Bio::SeqIO (sequence operations)

Third-Party Programs: RNAmmer (rRNA prediction)

Installation

clonning the repository

```
$ git clone https://github.com/fredericokremer/genix/
$ cd genix
```

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/`), 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.

Installing the dependences

To install the Python and Perl modules in Ubuntu/Debian-based computers we have a shell script `install_dependences.sh` in the directory `dependences/`, along with a "requeriments.txt" file for PIP.

```
$ cd dependence/
$ sudo bash install_dependences.sh
```

Using Genix

The user may also create an optimized protein database for a given taxonomic group of interest by using the `get_database.sh` by providing it's tax_id. First, a dataset of proteins associated to the tax_id Is downloaded from `Uniprot` and used to build a raw dataset, which may contain several redundances. `CD-HIT` (Li & Godzik 2006) is used to build a non-redundant dataset, which is paserd by an python script to identify the best annotation from each cluster based on the protein-score and the source from where the proteins were obtained (Uniprot-Swissprot or Uniprot-trEMBL). Finally, the refined dataset is used to generate the final protein `BLAST` database (Altschul et al. 1990; Camacho et al. 2009).

The user may also create an optimized protein database for a given taxonomic group of interest by using the `get_database.sh` by providing it's tax_id. First, a dataset of proteins associated to the tax_id Is downloaded from `Uniprot` and used to build a raw dataset, which may contain several redundances. CD-HIT (Li & Godzik 2006) is used to build a non-redundant dataset, which is paserd by an python script to identify the best annotation from each cluster based on the protein-score and the source from where the proteins were obtained (Uniprot-Swissprot or Uniprot-trEMBL). Finally, the refined dataset is used to generate the final protein BLAST database (Altschul et al. 1990; Camacho et al. 2009).

preparing the database

Genix requires a protein BLAST database to perform the annotation process. Although it is possible to run Genix using your own protein database, we strongly recommend the use of databases derived from UniProt, as they have a consistent and standartized FASTA header structure. Usually, genome annotation pipelines use generic subsets from Uniprot such as Swissprot and trEMBL, to perform the annotation, but both of these databases imply in drawnbacks for the annotation process, as Swissprot is manually curated but limited in terms of the number of sequences and covered taxons, while trEMBL has a larger number of sequences but many redundancies and erroneous data.

To take the best from both worlds (Swissprot and trEMBL) we have developed a script to use in the webserver version that performs the construction of the BLAST database for a specific taxonomic group using a "tax_id" using data retrieved from Uniprot, but with redundances removed and an improved annotation selected for each "cluster" by using aditional steps of refinament. The script, `get_database.sh`, which is available in the directory `scripts/`, receives 3 arguments: [1] a taxonomy identifier (tax_id) of the taxon of interest; [2] the name of the output database; [3] the identity threshold that must be used during the sequence clustering.

As an example, to use the script to prepare a database of *Leptospira* proteins names `lepto_db` using a identity threshold of 95% during the sequence clustering, just type:

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

Running the annotation

```
$ python genix_annotation.py

##############################################################
#////////// Genix Annotation //////////////////////////////#
# An automated pipeline for bacterial genome annotation    #
# Version: 0.4                                             #
# Reference: KREMER et al, 2016                            #
##############################################################

usage: genix_annotation.py [-h] -i INPUT [-o OUTPUT_DIR] [-t THREADS] -dbp
                           PROTEIN_DATABASE [-antie ANTIFAM_EVALUE]
                           [-blaste BLAST_EVALUE]
                           [-blastrnae BLAST_RNA_EVALUE]
                           [-nfernale INFERNAL_EVALUE] [-mg MIN_GAP]
```

```
                              [-gc {11,4}] [-sl SCAFFOLD_LABEL] [-lt LOCUS_TAG]
                              [-nst INSTITUTION] [-sp {BLAST}]
```

`-i / --input` : indicates the FASTA file that will be used as inputs.

`-o / --output_dir` : indicates the directory were the results will be saved.

`-t / --threads` : indicates the number of threads that will be used by BLAST, INFERNAL and HMMER.

`-dbp / --protein_database` : indicates the BLAST protein database to be used.

`-antie / --antifam_evalue` : indicates the e-value threshold that will be used when processing the results from the HMMER search against the Antifam database.

`-blaste / --blast_evalue` : indicates the e-value threshold that will be used when processing the results from the BLAST search against the protein database.

`-blastnrae / --blast_rna_evalue` : indicates the e-value threshold that will be used when processing the results from the BLAST search against the Rfam database.

`-nfernale / --invernal_evalue` : indicates the e-value threshold that will be used when processing the results from the INFERNAL search against the Rfam database.

`-gc` : The genetic code that will be used. Can be "11" (default bacteria genetic code) of "4" (*Mycoplasma*).

`-sl` : The prefix of the that will be used to index the scaffolds.

`-lt` : The `locus_tag` prefix of the that will be used to index the scaffolds.

`-mg / --min_gap` : minimum size of the runs of Ns to distinguish it from base-calling errors.

`-nst` : indicates the institution that must the indicated in the `protein_id` qualifiers of the CDSs.

`-sp` : the search program that will be used. In the current implementation Genix only supports BLAST, but we intend to provide suport for BLAT and USEARCH in future releases.

`-nst` : indicates the FASTA file that will be used as inputs.

Running the Example datasets

```
$ cd test_data/
$ bash run_annotation.sh
```

The code of this script has the following commands:

```
#!/usr/bin/env bash

##### THE DIRECTORIES

if [ ! -d database ]; then
    mkdir database
fi

if [ ! -d output ]; then
    mkdir output
fi

##### DATABASE PREPARATION

bash ../scripts/get_database.sh 561 database/e.coli 0.95

##### RUN THE ANNOTATION PIPELINE

python ../genix_annotation.py -i e.coli_K12.fasta -dbp database/e.coli \
       -o output/ --threads 4
```

This script prepares two directories: `database/` (to store the files from the BLAST database) and `output/` (to store the results from the annotation). To prepare the database, the `tax_id` "561" is used, which corresponds to the *Escherichia coli* taxon, and generates the database "e.coli" that is saved in the `database/`. To perform tha annotation, Genix takes as input the file `e.coli_K12.fasta`, which contains the genome sequence of *Escherichia coli* strain K-12, and uses the database generated in the previous step the the protein annotation. Finally, the intermediary files and the results will be saved to the directory `output/`. The argument `--threads 4` indicates that the programs from the packages `BLAST`, `INFERNAL` and `HMMER` will be executed using 4 threads.

WebAPI

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 “WebAPI.py” located in the directory “WebAPI_example” 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. As the API is based on HTTP requests, and uses JSON as the main data structure, it is possible to use also any "modern" language to access it.

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).

```
#!/usr/bin/env python

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,
            'rfam_blast':0.0005,
            'antifam_hmmer':0.0005,
            '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)})
job_id = json.loads(return_data.text)['job_id']
```

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.

Retrieving the status of a job

```
get_status_data = {
    'user_email' : 'fred.s.kremer@gmail.com',
    'user_password' : '1234',
    'job_id':job_id
}

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

Retrieving the result of a finished Job in GenBank format

The API is still under-development, so, for now, it is only possible to retrieve the results of an annotation in the GenBank format. To do it, follow the example bellow:

```
get_results_data = {
    'user_email' : 'fred.s.kremer@gmail.com',
    'user_password' : '1234',
    'job_id':job_id
}

result_data = requests.post("http://labbioinfo.ufpel.edu.br/cgi-bin/genix_api.py",
                            data={'mode':'get_results','data': json.dumps(
                                  get_results_data)})

genbank_file = json.loads(result_data.text)['file_genbank']
```

Webserver

Genix is also available as an webserver through the URL http://labbioinfo.ufpel.edu.br/genix.

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

References

Altschul, S.F. et al., 1990. Basic local alignment search tool. Journal of molecular biology, 215(3), pp.403–10.

Camacho, C. et al., 2009. BLAST+: architecture and applications. BMC bioinformatics, 10(1), p.421.

Eberhardt, R.Y. et al., 2012. AntiFam: a tool to help identify spurious ORFs in protein annotation. Database: the journal of biological databases and curation, 2012, p.bas003.

Eddy, S.R., 2011. Accelerated Profile HMM Searches. W. R. Pearson, ed. PLoS computational biology, 7(10), p.e1002195.

Griffiths-Jones, S. et al., 2003. Rfam: an RNA family database. Nucleic acids research, 31(1), pp.439–41.

Hyatt, D. et al., 2010. Prodigal: prokaryotic gene recognition and translation initiation site identification. BMC bioinformatics, 11, p.119.

Lagesen, K. et al., 2007. RNAmmer: consistent and rapid annotation of ribosomal RNA genes. Nucleic acids research, 35(9), pp.3100–8.

Laslett, D., 2004. ARAGORN, a program to detect tRNA genes and tmRNA genes in nucleotide sequences. Nucleic Acids Research, 32(1), pp.11–16.

Li, W. & Godzik, A., 2006. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinformatics (Oxford, England), 22(13), pp.1658–9.

Lowe, T.M. & Eddy, S.R., 1997. tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. Nucleic acids research, 25(5), pp.955–64.

Nawrocki, E.P., Kolbe, D.L. & Eddy, S.R., 2009. Infernal 1.0: inference of RNA alignments. Bioinformatics (Oxford, England), 25(10), pp.1335–7.