Antimicrobial Resistance Gene Database Integration Toolkit

User manual

Contents

Overview

The Antimicrobial Resistance Gene Data Integration Toolkit (ARGDIT) consists of two main tools and three utilities for users to perform data validation and integration on antimicrobial resistance gene (ARG) databases. Basically it allows users to validate the fidelity of an ARG (or other coding sequence/protein) database to the coding sequence/protein information from the NCBI repositories, and to merge multiple validated databases into a single ARG database. It also supports automated re-annotating the output ARG sequences with NCBI sequence information, as well as sequence classification (i.e. predicting the class labels of the database sequences) according to a schema database, which is another ARG database containing classified sequences.

Note that although the default translation table used is for bacteria, **other translation tables can also be used for non-bacterial coding sequence databases** by specifying a different genetic code. Refer to the usage details of the two tools below.

Main tools:

- ARG database validation tool (check_arg_db.py)
- ARG database integration tool (merge_arg_db.py)

Utilities:

- Database sequence replacement utility (replace_db_seqs.py)
- UniProt identifier to NCBI protein accession number conversion utility (convert_id_uniprot_to_ncbi.py)
- Database diff utility (seq_db_diff.py)

Database eligibility

In order to use the data validation and integration tool, the ARG (or other coding/protein sequence) database must be

1. In FASTA format
2. Every FASTA sequence header must contain an NCBI nucleotide/protein accession number. Uniprot ID is an alternative for protein accession number for protein sequence database (by converting the Uniprot IDs to protein NCBI accession numbers with the conversion utility provided)
3. Every sequence must either be a coding sequence (i.e. will be translated to protein product) or a protein sequence
4. ARG/Sequence class information, if any, must occupy at least one individual field in the sequence headers, in which all the fields are separated by the "|" symbol

Pre-requisites

The followings must be installed for the core ARGDIT operations:

1. Python version 3.5 or higher
2. BioPython version 1.70 or higher

If sequence classification or class outlier sequence detection is required, then the followings must be installed:

1. MUSCLE version 3.8.31 or higher
2. OD-Seq
3. HMMER3 version 3.1b2 or higher

Installation

No installation is required. Make sure all the third-party software in pre-requisites are in the system path.

In order to access the NCBI repositories, users must provide their own contact email addresses along with their access requests. Fill in your contact email address under the "Entrez" section in the configuration file (config.ini):

```
[Entrez]
Email = (your contact email address)
```

Important notice

All data retrieval of NCBI repositories are performed through NCBI Entrez Programming Utilities. Before using ARGDIT it is very important for every user to read its guidelines and requirements (https://www.ncbi.nlm.nih.gov/books/NBK25497/#chapter2.Usage_Guidelines_and_Requiremen) and avoid overwhelming the NCBI servers according to the guidelines. Based on these requirements, users are required to provide their contact email addresses (see the Installation section) so that NCBI may attempt contact before **blocking the abusing access**. Although this email address is intended for the software developers, it is more appropriate for the users to fill in their own so that they can be notified by NCBI.

Configurations

The configurable parameters for ARGDIT can be found in the configuration file "config.ini". These parameters are categorized into different sections listed in the table below:

Section	**Parameter**	**Description**	**Default**
ARGDIT	FastaHeaderFieldSeparator	Field separator in the FASTA sequence header	|
	OperationalFieldSeparator	FASTA sequence header field separator to use during program execution; replaces original field separator (specified by FastaHeaderFieldSeparator) during operation	__
Sequence classification check	MinSequenceCount	Minimum number of sequences for a sequence class to be validated or used for classification	3
	BootstrapFactor	Determines the number of bootstrap iterations for sequence outlier detection according to the formula: No. of bootstrap iterations = No. of sequences in a class × bootstrap factor	1000
Entrez	Email	User's contact email address for the Entrez utilities	
Translate	DefaultGeneticCode	Default translation table used	11

Sequence header field indexing

One-based indexing is applied to index the sequence fields. Assuming the use of the default FASTA sequence header field separator ("|"), for the sample header at the end of this section, the third field is "beta-lactamase_CTX-M-134", and the fourth field is empty string "". The fields can also be indexed from the last field back to the first field, with the last field indexed as -1, the second last field indexed as -2, and so on. For example, the field with index -4 in the sample header is "Escherichia_coli". Note that due to input limitation the negative sign "-" is replaced by "~" in the tool input argument.

Multiple fields can be extracted from the header by slice. For the sample header, by specifying 1-2 the extracted information is "JX896165|blaCTX-M-134", while "1-876|876|complete" is extracted with the slice ~1–~3.

```
JX896165|blaCTX-M-134| beta-lactamase_CTX-M-134|      |Escherichia_coli| 1-876| 876| complete
    1          2                 3                 4          5           6     7      8
   -8         -7                -6                -5         -4          -3    -2     -1
```

Usage

Database validation tool

Command

```
./check_arg_db.py [optional arguments] seq_db_path
```

Mandatory argument

`seq_db_path`	nucleotide/protein database FASTA file path

Optional arguments

`-f/--fields FIELD_NUMS`	sequence class label field numbers FIELD_NUMS for class outlier sequence detection, e.g. -f 4-5, -f ~1-~3
`-r/--refine`	export refined DNA sequences
`-c/--geneticcode GENETIC_CODE`	genetic code to specify which translation table to be used
`-e/--exportlog`	export validation results and process log
`-h/--help`	show help message and exit

Description

check_arg_db.py performs ARG database validation. The --refine option allows the tool to trim at most two spurious bases before the start codon or after the stop codon, and export the trimmed sequences into an individual file specified by the tool. To perform ARG class outlier sequence detection, specify the ARG class fields after the --fields option. For example, the hierarchical ARG class information can be extracted from MEGARes database by "-f ~1-~3". The --geneticcode option overrides the default genetic code specified in the configuration file. The genetic code represents the translation table to be used when translating the DNA sequences. As a result, sequence databases for organisms other than bacteria can also be validated. Refer to here for the genetic codes representing different translation tables. The validation results and process log are printed to stdout (i.e. screen) by default, and by specifying the --exportlog option they will be sent to a .log file in the same directory as the database file.

Database integration tool

Command

```
./merge_arg_db.py [optional arguments] -o OUTPUT_SEQ_DB_PATH seq_db_paths
```

Mandatory arguments

`-o OUTPUT_SEQ_DB_PATH`	specify the output database file path OUTPUT_SEQ_DB_PATH
`seq_db_paths`	nucleotide/protein database FASTA file paths

Optional arguments

`-s/--schema SCHEMA_DB_PATH FIELD_NUMS`	specify the schema database SCHEMA_DB_PATH and class label field numbers FIELD_NUMS to perform sequence class prediction
`-a/--annotate`	perform automated re-annotation using NCBI repository information
`-p/--protein`	export protein sequences
`-r/--redundant`	allow redundant sequences
`-c/--geneticcode GENETIC_CODE`	genetic code to specify which translation table to be used
`-e/--exportlog`	export integration results and process log
`-h/--help`	show help message and exit

Description

merge_arg_db.py performs integration of multiple ARG databases. The --annotate option performs re-annotation of the sequences in the output database. By specifying the schema database file path and the ARG class fields after the --schema option, the class labels of the output sequences will be predicted. However, note that the protein sequences of the schema database are not validated here, so it is advised to validate them using the validation tool. The --geneticcode option overrides the default genetic code specified in the configuration file. The genetic code represents the translation table to be used when translating the DNA sequences. As a result, sequence databases for organisms other than bacteria can also be consolidated. Refer to here for the genetic codes representing different translation tables. By default only non-redundant sequences are exported, but this can be overridden with the --redundant option. The tool provides the --protein option to translate all DNA sequences to protein sequences.

Sequence replacement utility

Command

```
./replace_db_seqs.py [optional argument] seq_db_path replace_seq_file_path
output_seq_db_path
```

Mandatory arguments

`seq_db_path`	nucleotide/protein database FASTA file path
`replace_seq_file_path`	FASTA file path for replacement sequences
`output_seq_db_path`	output database file path

Optional argument

`-h/--help`	show help message and exit

Description

By matching identical FASTA headers of the sequences, this utility replaces the sequences in the database FASTA file with those in the replacement sequence file. The database sequences, no matter replaced or not, are exported to the output database file specified by the user.

Uniprot ID conversion utility

Command

```
./convert_id_uniprot_to_ncbi.py [optional argument] seq_db_path
output_seq_db_path
```

Mandatory arguments

`seq_db_path`	FASTA file path for protein database with Uniprot IDs
`output_seq_db_path`	output file path for protein database with converted NCBI protein accession no.

Optional argument

`-h/--help`	show help message and exit

Description

This tool queries the UniProt database for the Uniprot ID to NCBI protein accession number mappings, and then replaces the UniProt IDs in the sequence headers by the mapped protein accession numbers. The processed sequences are exported to the output database file specified by the user.

<u>Database diff utility</u>

Command

```
./seq_db_diff.py [optional arguments] seq_db_path old_seq_db_path
```

Mandatory arguments

`seq_db_path`	nucleotide/protein database FASTA file path
`old_seq_db_path`	database FASTA file path for the previous release

Optional arguments

`-m/--mode {h,s,b}`	diff mode (h:header; s:sequence; b:both [default])
`-o/--old`	export also sequences found in old database only
`-f/--forward`	do not check reverse complement nucleotide sequences
`-h/--help`	show help message and exit

Description

This utility compares the sequence database with its previous version according to the diff mode specified. Header (h) mode performs diff by comparing sequence annotation header only (hence ignoring residue sequence); sequence (s) mode performs diff by comparing residue sequence only (hence ignoring sequence annotation header); both (b) mode performs diff by comparing both annotation header and residue sequence. A FASTA file storing all identical sequences (according the diff mode) is generated, as well as another FASTA file storing those that appear in the current database only. The --old option can export a FASTA file storing sequences that appear in the old database only. By default, when the diff mode is other than header mode and the database sequences are DNA sequences, residue sequence comparison is also performed with reverse complement (i.e. a DNA sequence S1 is compared with another DNA sequence S2 as well as the reverse complement of S2). However, the reverse complement comparison can be suspended by the --forward option.

Known issues

It is sometimes (but not often) possible to have incomplete data retrieval from NCBI repositories due to server-side issues such as heavy workload. This means information for some nucleotide/protein accession numbers cannot be retrieved at the moment; the outcome is like these accession numbers are not present in the NCBI repositories. When many sequences are spuriously reported as having their accession numbers not found and/or sequence mismatches, it is advised to try using the database validation and the integration tools later.

As a workaround, try to reduce the number of accession no. included in each NCBI repository query request. By default each query request consists of at most 3000 unique accession no., and this setting can be adjusted by modifying both constants EPOST_FT_BATCH_SIZE and EPOST_SEQ_BATCH_SIZE in the library script EntrezDBAccess.py (in folder ArgditLib) to lower values such as 500 or 1000 (default is 3000). Note that this workaround may not be effective all the time.