ncfp Documentation

Release 0.1.0dev

Leighton Pritchard

Jun 02, 2019

Contents:

If you're feeling impatient, please head over to the *QuickStart Guide*

CHAPTER 1

Description

`ncfp` is a program and Python package that, given a set of protein sequences with appropriate identifier values, retrieves corresponding coding nucleotide sequences from the `NCBI nucleotide` databases. This is useful, for instance, to help backthread coding sequences onto protein alignments for phylogenetic analyses.

CHAPTER 2

Reporting problems and requesting improvements

If you encounter bugs or errors, or would like to suggest ways in which `ncfp` can be improved, please raise a new issue at the `ncfp` GitHub issues page.

If you'd like to fix a bug or make an improvement yourself, contributions are welcomed, and guidelines on how to do this can be found at the *Contributing* documentation page.

CHAPTER 3

Getting started

To get started, please read the documentation, below:

3.1 About ncfp

ncfp is a program and accompanying Python package (ncbi_cds_from_protein) that, given a set of protein sequences with appropriate identifier values, retrieves corresponding coding nucleotide sequences from the NCBI nucleotide databases. This is useful, for instance, to help backthread coding sequences onto protein alignments for phylogenetic analyses.

3.1.1 How ncfp works

1. The ncfp program accepts input FASTA sequence files describing protein sequences. These describe a set of queries that will be used to obtain corresponding nucleotide coding sequences, if possible.

Input sequence formats Please see *Input sequence formats*.

2. An SQLite cache database is created. This will hold information about the query sequences, and about the data downloaded from NCBI using the input sequence data as queries. The cache enables recovery from interrupted jobs, and reuse of data without needing to transfer across the network again by interrogating the SQLite database directly.

Cache directory By default, the cache is created in the hidden subdirectory .ncfp_cache, but any location can be specified using the -d or --cachedir arguments.

Cache filename By default, the cache has a filestem reflecting the date and time that ncfp is run, but this can be changed using the -c or --cachestem arguments.

3. The Biopython Entrez library is used to make a connection to the NCBI sequence databases. Using this connection, the program identifies nucleotide database coding sequence entries that are related to each input protein sequence. The relationship is determined on the basis of either the NCBI accession, or the UniProt gene identifier, depending on the input file.

Batched downloads By default, ncfp makes queries and downloads sequences in batches of 100. The batch size can be controlled using the arguments -b or --batchsize

Retry attempts Sometimes network connections are flaky, so by default `ncfp` will try each request up to 10 times. The number of retries can be controlled with the `-r` or `--retries` arguments.

4. If the results of each `NCBI` query are not already present in the cache, they are downloaded and recorded in the cache as header information. Some specific data are extracted, (sequence length, taxonomy, etc.)

5. The shortest available complete coding sequence[1] that recapitulates each input protein sequence is identified. If the sequence is not already present in the cache, it is downloaded.

6. Pairs of protein and corresponding coding sequences are written to two files: one for nucleotide sequences, and one for protein sequences. Sequences are written to each file in the same order, so they can be used for backtranslation with a tool such as T-Coffee. If any proteins could not be matched to their coding sequence at `NCBI`, they are written to a third file.

Output filenames The filestem for the paired protein and coding sequence files is always suffixed by `_aa` or `_nt`, depending on the type of sequence being written. The filestem is `ncfp` by default, but this can be controlled with the `--filestem` argument.

Skipped sequences filename By default, the protein sequences for which a coding sequence could not be found are written to the `skipped.fas` file. An alternative path can be provided with the `--skippedfile` argument.

3.2 QuickStart Guide

3.2.1 Installation

Using `conda`

`ncfp` is available through the `bioconda` channel of Anaconda:

```
conda install -c bioconda ncfp
```

From source

At the command-line, use `git` to clone the current version of the `ncfp` repository:

```
git clone git@github.com:widdowquinn/ncfp.git
```

Change to the newly-created `ncfp` subdirectory:

```
cd ncfp
```

Install the package and program, using the `setup.py` file:

```
python setup.py install
```

(other installation methods can be found on the *Installation* page)

3.2.2 `ncfp` Example

To see options available for the `ncfp` program, use the `-h` (help) option:

[1] We require the complete coding sequence, but if we can use a shorter sequence than the complete genome, we do to save bandwidth.

```
ncfp -h
```

In the `ncfp/tests/test_input/sequences` subdirectory there is a file called `input_ncbi.fasta`. This contains a number of protein sequences in FASTA format. The identifier for each sequence in this file is a valid NCBI sequence identifier.

Using `ncfp`, to obtain a corresponding nucleotide coding sequence for each protein, issue the following command (substituting your own email address, where indicated):

```
ncfp tests/test_input/sequences/input_ncbi.fasta \
     example_output \
     my.name@my.domain
```

You should see progress bars appear for processing of the input protein, sequences, searching those sequences against the remote `NCBI` databases, then retrieving the corresponding sequence identifiers, GenBank headers and finally the full GenBank records.

On completion, a list of the recovered sequences will be presented, and the directory `example_output` will be created, with the following contents:

```
$ tree example_output/
example_output/
├── ncfp_aa.fasta
└── ncfp_nt.fasta
```

The two files should contain corresponding amino acid and nucleotide sequences:

```
$ head example_output/*.fasta
==> example_output/ncfp_aa.fasta <==
>XP_004520832.1 kunitz-type serine protease inhibitor homolog dendrotoxin I-like␣
↪[Ceratitis capitata]
MRTKFVLVFALIVCVLNGLGEAQRPAHCLQPHPQGVGRCDMLISGFFYNSERNECEQWTE
EGCRVQGGHTYDFKEDCVNECIEIN
>XP_017966559.1 PREDICTED: kunitz-type serine protease inhibitor homolog dendrotoxin␣
↪I-like [Drosophila navojoa]
MKFILLLACLCVYVATLEAQRPPCKGIVPPWLTNCVGGKNEGRGNLRSCARNANSRMWWY
DSRSRSCKKMAYKGCGGNRNRYCTREACRRACRRRN
>XP_017841791.1 PREDICTED: kunitz-type serine protease inhibitor homolog dendrotoxin␣
↪K-like [Drosophila busckii]
MKVCLILSALVLQYIVFVNAEGCPLRPAEQNCQSSRNVGVSSYSNCILTKRLMWYYNPTI
RDCLPLDFRGCGGNGNRYCSLKDCQQSCKHT
>XP_017046608.1 PREDICTED: kunitz-type serine protease inhibitor homolog dendrotoxin␣
↪I [Drosophila ficusphila]

==> example_output/ncfp_nt.fasta <==
>XP_004520832.1 coding sequence
ATGAGAACTAAATTTGTTTTGGTATTCGCGCTCATTGTTTGTGTACTCAACGGTTTAGGT
GAAGCGCAAAGACCAGCACATTGCTTACAACCACATCCACAAGGAGTTGGCCGTTGTGAT
ATGCTTATCAGTGGTTTCTTCTATAACTCGGAGCGTAATGAGTGCGAGCAATGGACAGAG
GAGGGCTGCCGTGTGCAGGGTGGGCACACATACGATTTCAAAGAAGATTGTGTAAATGAG
TGCATTGAAATTAATTAA
>XP_017966559.1 coding sequence
ATGAAATTCATTCTGCTCCTCGCTTGTCTCTGCGTCTACGTGGCCACCCTTGAGGCTCAG
CGACCCCCTTGCAAGGGAATAGTGCCTCCATGGTTGACCAATTGTGTTGGAGGCAAGAAC
GAGGGCAGGGGTAACCTTCGCTCGTGCGCCAGGAACGCGAATTCCAGAATGTGGTGGTAT
```

3.3 Requirements

To use `ncfp` you will need:

Operating System

- macOS or Linux

Python

- Python version 3.5 or higher

Python Packages

- biopython
- tqdm

3.4 Installation

`ncfp` can be installed in any of several ways.

3.4.1 Using `pip`

The most recent release of `ncfp` is available at the PyPI warehouse, and can be installed using `pip`:

```
pip install ncfp
```

3.4.2 Using `bioconda`

`ncfp` is available through the bioconda channel of the conda package management system. To install `ncfp`, you will need Anaconda or miniconda, and to set up the `bioconda` channel. Then you can use the `conda install ncfp` command to install the package:

```
conda install -c bioconda ncfp
```

Alternatively:

```
conda config --add channels defaults
conda config --add channels conda-forge
conda config --add channels bioconda
conda install ncfp
```

3.4.3 From source (most recent release)

`ncfp` releases are available at the GitHub releases page. To install from source:

1. Download the *.tar.gz* or *.zip* file containing the package (click the link, use `curl` or `wget`).
2. Uncompress the archive.
3. Change to the newly-expanded directory.
4. Install Python module requirements with `pip`.

5. Install `ncfp` with `setup.py`

```
$ wget https://github.com/widdowquinn/ncfp/archive/v0.1.0.tar.gz
$ tar -zxvf v0.1.0.tar.gz
$ cd ncfp
$ pip install -r requirements.txt
$ python setup.py install
```

3.4.4 From source (bleeding edge)

To get the very latest development version of `ncfp`, you can clone the repository from the GitHub project page

1. Clone the repository from `git@github.com:widdowquinn/ncfp.git`
2. Change to the repository root directory.
3. Install Python module requirements with `pip`.
4. Install `ncfp` with `setup.py`

```
$ git clone git@github.com:widdowquinn/ncfp.git
$ cd ncfp
$ pip install -r requirements.txt
$ python setup.py install
```

3.5 Basic Use

Given a set of protein sequences in the file `<INPUT>.fasta` as input, the path to an output directory as `<OUTPUT>`, and the user's email address[1] as `<EMAIL>`, the following command will query the NCBI databases for nucleotide coding sequences corresponding to the input, and write the results to files in `<OUTPUT>`:

```
ncfp <INPUT>.fasta <OUTPUT> <EMAIL>
```

The output directory `<OUTPUT>` will contain at least two files: `<OUTPUT>/ncfp_aa.fasta` and `<OUTPUT>/ncfp_nt.fasta`. The `<OUTPUT>/ncfp_aa.fasta` file will contain sequences for which a corresponding coding sequence could be found, and `<OUTPUT>/ncfp_nt.aa` will contain those coding sequences.

The sequences in `<OUTPUT>/ncfp_nt.aa` are trimmed and validated by `ncfp` to produce a conceptual translation identical to the corresponding input protein sequence.

Any protein sequences for which a partner nucleotide coding sequence could not be found will be written to the file `<OUTPUT>/skipped.fas`

3.5.1 Input sequence formats

Input protein sequences must be provided in FASTA format, and `ncfp` expects input sequence headers to take one of two forms: "NCBI" or "UniProt". By default, `ncfp` expects sequences to be in NCBI format:

```
ncfp <INPUT>.fasta <OUTPUT> <EMAIL>
```

For sequence input in UniProt format, one of the `-u` or `--uniprot` options must be used, e.g.

[1] The user's email address is passed to NCBI to enable them to monitor use of their service and provide support

```
$ ncfp -u <INPUT>.fasta <OUTPUT> <EMAIL>
$ ncfp --uniprot <INPUT>.fasta <OUTPUT> <EMAIL>
```

NCBI header format

In NCBI header format, the sequence identifier is expected to correspond to a valid NCBI protein sequence accession, e.g.

```
>XP_004520832.1 kunitz-type serine protease inhibitor homolog dendrotoxin I-like␣
↪[Ceratitis capitata]
MRTKFVLVFALIVCVLNGLGEAQRPAHCLQPHPQGVGRCDMLISGFFYNSERNECEQWTEEGCRVQGGHT
YDFKEDCVNECIEIN
```

If a coding sequence is identified successfully, the output nucleotide sequence header will have the same accession as a sequence identifier, e.g.

```
>XP_004520832.1 coding sequence
ATGAGAACTAAATTTGTTTTGGTATTCGCGCTCATTGTTTGTGTACTCAACGGTTTAGGT
GAAGCGCAAAGACCAGCACATTGCTTACAACCACATCCACAAGGAGTTGGCCGTTGTGAT
ATGCTTATCAGTGGTTTCTTCTATAACTCGGAGCGTAATGAGTGCGAGCAATGGACAGAG
GAGGGCTGCCGTGTGCAGGGTGGGCACACATACGATTTCAAAGAAGATTGTGTAAATGAG
TGCATTGAAATTAATTAA
```

UniProt header format

In UniProt header format, the sequence description string is expected to correspond to a UniProt download and contain the `GN` gene identifier key:value pair, e.g.

```
>tr|A0A1V9Y7A7|A0A1V9Y7A7_9STRA Lon protease homolog OS=Thraustotheca clavata␣
↪GN=THRCLA_11583 PE=3 SV=1
MYRASSKVTSAHNDGIWSTVWTSRNQIISGSLDEVVKSWDASSSEDNAILPVVKQFPGHV
LGTLAVTATKDGRKAATSSLDCQVRILNLESGGIEKTIDTGAGESWQLVYSPDDTFIATG
SQQSKINLINLEQEKIVNSIPVDGKFILAVAYSPDGKHLACGTFEGIVAIYDVETGKQVQ
KYQDRAKPVRSISYSPDGSFLLAASDDMHVNIYDVLHSSLVGSVSGHISWILSVACSPDG
```

If a coding sequence is identified successfully, the output nucleotide sequence header will have the gene accession as its sequence identifier, e.g.

```
>THRCLA_11583 coding sequence
ATGTACCGCGCCTCGTCCAAAGTAACGTCGGCTCATAATGATGGAATCTGGAGTACTGTC
TGGACAAGCCGCAATCAAATCATAAGTGGATCTTTGGATGAAGTGGTCAAGAGCTGGGAT
GCGAGTAGTTCCGAGGACAATGCGATTTTGCCTGTTGTCAAGCAATTTCCAGGCCACGTT
CTAGGCACACTGGCAGTGACTGCAACGAAAGATGGTCGAAAAGCTGCTACATCGTCTTTA
```

Stockholm domain format

UniProt and other sources use Stockholm format to indicate that an amino acid sequence represents a portion of a protein (such as a domain). `ncfp` can recognise this format and trim the coding sequence to correspond only to the specified region of the protein.

Stockholm format domains are indicated by the syntax `/<start>-<stop>` immediately following the sequence identifier in FASTA format, e.g.

```
>tr|B7G6L2|B7G6L2_PHATC/43-112 [subseq from] Predicted protein OS=Phaeodactylum␣
↪tricornutum (strain CCAP 1055/1) GN=PHATRDRAFT_48282 PE=4 SV=1
-----------------------------SLCV-EVAGA-SQD---DGASIFQGDCN-dG
NKHQVFDFipaPG---TdsgFHRIRA--SHSN-KCLGVADGAL--APG-AEVVQ-
```

To restrict the coding sequence to the region indicated in Stockholm format, pass either the `-s` or `--stockholm` option, e.g.

```
$ ncfp -u -s <INPUT>.fasta <OUTPUT> <EMAIL>
$ ncfp --uniprot --stockholm <INPUT>.fasta <OUTPUT> <EMAIL>
```

The output nucleotide sequence does not preserve the Stockholm format location information in the output, nor does it preserve sequence gap symbols:

```
>PHATRDRAFT_48282 coding sequence
TCGCTCTGCGTGGAGGTGGCTGGAGCGAGCCAAGACGACGGGGCCTCCATATTTCAAGGG
GATTGTAATGACGGAAACAAGCATCAAGTCTTCGACTTCATTCCTGCTCCCGGTACAGAC
AGCGGTTTTCATCGAATTCGAGCCTCGCACTCCAACAAGTGCCTTGGCGTGGCTGATGGG
GCTTTAGCACCTGGAGCTGAGGTAGTGCAA
```

3.6 Examples

3.6.1 NCBI format input sequences - no introns

The command below[1] identifies coding sequences from NCBI format[2] input for nine proteins that do not contain introns.

```
ncfp tests/test_input/sequences/input_ncbi.fasta \
    tests/test_output/ncbi dev@null.com -v
```

3.6.2 UniProt format input sequences - no introns

The command below[1] identifies coding sequences from UniProt format[3] input for ten proteins that do not contain introns. The `-u` or `--uniprot` argument is required to specify that the input sequences are UniProt format, otherwise an error is thrown.

```
ncfp -u tests/test_input/sequences/input_uniprot.fasta \
    tests/test_output/uniprot dev@null.com -v
```

3.6.3 UniProt/Stockholm input sequences - no introns

The command below[1] identifies coding sequences from UniProt format[3] input for 57 amino acid sequences specifying regions of a protein in Stockholm notation[4]. The `-u` or `--uniprot` argument is required to specify that the input sequences are UniProt format, and the `-s` or `--stockholm` arguments are required to tell `ncfp` to parse the region locations.

[1] The `-v` option shows verbose output in `STDOUT`.
[2] The sequence identifier in the FASTA header is a valid NCBI protein accession.
[3] The sequence description in the FASTA header contains a valid `GN=<accession>` gene identifier.
[4] The sequence identifier in the FASTA header ends with a Stockholm format region definition, e.g. `/47-134`.

```
ncfp -us tests/test_input/sequences/input_uniprot_stockholm.fasta \
    tests/test_output/uniprot_stockholm dev@null.com -v
```

3.6.4 Human sequences - isoforms and intron/exon structure

The command below[1] identifies coding sequences from NCBI format[3] input for four human proteins with intron/exon structure, including three isoforms of the same protein from the same locus (GPR137: `NP_001164351.1`, `NP_001164352.1`, and `XP_005274161.1`).

```
ncfp tests/test_input/sequences/human.fasta \
    tests/test_output/human dev@null.com -v
```

3.6.5 Logging

Verbose output can be written persistently to a logfile using the `-l` or `--logfile` argument and specifying the path to which the logfile should be written. An example is given in the command below.

```
ncfp tests/test_input/sequences/human.fasta \
    tests/test_output/logging dev@null.com \
    -l tests/test_output/logging/human.log
```

3.6.6 Specifying the cache location

By default a new cache database is created every time that `ncfp` is run, in the `.ncfp_cache` hidden subdirectory. The default cache database filename is `ncfpcache_YYYY-MM-DD-HH-MM-SS.sqlite3`, indicating the time that the command was run. This location and naming convention can be overridden with the `-d`/`--cachedir` and `-c`/`--cachestem` arguments, as in the command below.

```
ncfp tests/test_input/sequences/human.fasta \
    tests/test_output/caches dev@null.com \
    -d tests/test_output/caches \
    -c ncfp_cache
```

3.6.7 Reusing an existing cache

To avoid unnecessary bandwidth/`NCBI` queries, an existing cache database can be used. The location of the cache is specified with the `-d`/`--cachedir` and `-c`/`--cachestem` arguments, and the `--keepcache` option must be specified. If the specified location does not contain a cache database, one is created. For example:

```
ncfp tests/test_input/sequences/human.fasta \
    tests/test_output/caches dev@null.com \
    -d tests/test_output/caches \
    -c ncfp_cache
```

will create a cache at `tests/test_output/caches/ncfp_cache.sqlite3`, and

```
ncfp tests/test_input/sequences/human.fasta \
    tests/test_output/caches dev@null.com \
    -d tests/test_output/caches \
    -c ncfp_cache \
```

(continues on next page)

(continued from previous page)

```
    --filestem cached \
    --keepcache
```

will reuse the cachefile without making new queries at NCBI, and write the output to cached_aa.fasta and cached_nt.fasta[5].

3.7 Testing

ncfp tests are implemented in the Nose framework[1].

To run all tests locally, please issue the command:

```
nosetests -v
```

3.7.1 Obtaining test coverage information

The Nose framework integrates with the coverage.py module, and an account of the extent of test coverage can be obtained by running the following command:

```
nosetests -v --with-coverage \
    --cover-package=ncbi_cds_from_protein \
    --cover-html
```

3.8 Contributing

3.8.1 Reporting bugs and errors

If you find a bug, or an error in the code or documentation, please report this by raising an issue at the GitHub issues page for ncfp

3.8.2 Contributing code or documentation

We gratefully accept code contributions, if you would like to fix a bug, improve documentation, or extend ncfp. To make everyone's lives easier in this process, we ask that you please follow the procedure below:

1. Fork the ncfp repository under your account at GitHub.
2. Clone your fork to your development machine.
3. Create a new branch in your forked repository with an informative name, e.g. `git checkout -b fix_issue_107`.
4. Make the changes.
5. Run the repository tests (see the *Testing* documentation for more details).
6. If the tests pass, push the changes to your fork, and submit a pull request against the original repository.
7. Indicate one of the ncfp developers as an assignee in your pull request.

[5] The --filestem argument changes the filestem of the output nucleotide and amino acid sequence files.
[1] We are aware that nosetests is in maintenance mode, and a move to py.test is planned.

3.8.3 Suggestions for improvement

If you would like to make a suggestion for how we could improve `ncfp`, we welcome contributions at the GitHub issues page.

3.9 Licensing

Unless otherwise indicated, all code is subject to the following agreement:

(c) The James Hutton Institute 2017-2018 Author: Leighton Pritchard

Contact: leighton.pritchard@hutton.ac.uk

Address: Leighton Pritchard, Information and Computational Sciences, James Hutton Institute, Errol Road, Invergowrie, Dundee, DD6 9LH, Scotland, UK

3.9.1 The MIT License

3.10 Indexing and Search

- genindex
- modindex
- search