Codons

Release 1

Andrew Philip Freiburger

May 06, 2022

CONTENTS

The Codons module is a lightweight tool for a) conducting transcription and translation of genetic sequences, either from a FASTA formated file or a string; b) making and reading FASTA or multi-FASTA files of genetic and protein sequences; and c) conducting BLAST searches of protein and nucleotide sequences. Example Notebooks of these features are offered in the "examples" directory of the Codons GitHub repository.

CHAPTER

ONE

INSTALLATION

The following command installs Codons in a command prompt/terminal environment:

```
pip install codons
```

CHAPTER

TWO

CONTENTS

2.1 Codons API

2.1.1 Codons()

The data environment, in a Python IDE, is defined:

```
import codons
cd = codons.Codons(sequence = None, codons_table = 'standard', amino_acids_form = 'one_
↪letter', hyphenated = None, verbose = False, printing = True)
```

- *sequence* `str`: optionally specifies the genetic sequence that will be processed through subsequent functions. The sequence can alternatively be provided ad hoc to each function.
- *codons_table* `str`: specifies the framework for translating codons into amino acids, where the standard translation table is used by default.
- *amino_acids_form* `str`: specifies whether the amino acid `full_name`, `three_letter`, or `one_letter` nomenclature will be used in the protein sequence.
- *hyphenated* `bool`: specifies whether amino acid residues of the protein sequence are delimited by hyphens, where `None` defaults to `True` for `amino_acids_form = full_name` and `amino_acids_form = three_letter` and `False` for `amino_acids_for = one_letter`.
- *verbose* & *printing* `bool`: specifies whether troubleshooting information or results will be printed, respectively.

read_fasta()

A FASTA-formatted file is parsed into the constituent sequences and descriptions:

```
sequences, descriptions, fasta_file = cd.read_fasta(fasta_path = None, fasta_link =␣
↪None):
```

- *fasta_path* `str`: The path to a FASTA file that will be loaded, parsed, and returned.
- *fasta_link* `str`: The URL link to a FASTA file that will be imported, parsed, and returned.

Returns:

- *sequences* & *descriptions* `list`: The sequences and descriptions that are contained within the FASTA file.
- *fasta_file* `str`: The original FASTA file as a string.

make_fasta()

A simple function that constructs, returns, and optionally exports a FASTA-formatted file from the parameterized description and sequence:

```
fasta_file = cd.make_fasta(sequence, description = 'sequence', export_path = None):
```

- *sequence* `str`: The genetic or protein sequence that will constitute the FASTA file.
- *description* `str`: A description of the sequence that will be the first line of the FASTA file, which is appended with the length of each respective sequence in the FASTA file.
- *export_path* `str`: The path to which the FASTA file will be exported, where `None` specifies that the file will not be exported.

Returns:

- *fasta_file* `str`: The generated FASTA file.

complement()

A simple function that constructs and returns the complementary strand for a parameterized genetic sequence:

```
fasta_file = cd.complement(sequence = None, dna = True):
```

- *sequence* `str`: The genetic sequence for which the complemenentary strand will be determined.
- *dna* `bool`: specifies whether the parameterized strand is DNA or RNA.

Returns:

- *complementary_strand* `str`: The complementary genetic sequence.

transcribe()

A genetic sequence is converted from DNA -> RNA, or RNA -> DNA, where the directionality of the conversion is automatically listed in the FASTA description:

```
transcribed_sequence = cd.transcribe(sequence = None, description = '', fasta_path =␣
↪None, fasta_link = None)
```

- *sequence* `str`: The genetic seqeuence that will be transcribed. The sequence is case-insensitive, and can even possess line numbers or column-spaces, which the code ignores. The parameterization of `None` defaults to the sequence that was initially loaded into the `Codons` object.
- *description* `str`: A description of the genetic sequence that will be added to the FASTA-formatted output of the function.
- *fasta_path* & *fasta_link* `str`: The path or URL link to a FASTA file that will be transcribed.

Returns:

- *transcribed_sequence* `str`: The translated sequence.

find_start()

Determines the index of the next start codon:

```
index = cd.find_start(i, sequence):
```

- *i* `int`: The string index of the sequence after which a start codon will be searched.
- *sequence* `str`: The genetic sequence in which the start codon will be search.

Returns:

- *index* `int`: The index of the next start codon.

translate()

A genetic sequence is translated into proteins that is coded by the genetic code:

```
proteins = cd.translate(sequence = None, fasta_path = None, fasta_link = None, organism␣
↪= 'bacteria', start_codons = None, all_possible_proteins = False,
                        open_reading_frames = True,  filter_protein_size = 30, sense_
↪strand_translation = False)
```

- *sequence* `str`: The genetic sequence , of either DNA or RNA, that will be translated into a protein sequence. The sequence is case-insensitive, and can even possess line numbers or column-spaces, which the code ignores. The absence of a passed sequence executes the sequence that is loaded into the `Codons` object.
- *fasta_path* & *fasta_link* `str`: The path or URL link to a FASTA file that will be translated.
- *organism* `str`: specifies the type of organism whose genome is being translated, which informs which set of default start codons will be used: `['ATG', 'AUG', 'GTG', "GUG"]` for `bacteria` or `['ATG', 'AUG']` for `virus`.
- *start_codons* `list`: specifies the start codons that will be used, where `None` defaults to those from the `organism` selection.
- *all_possible_proteins* `bool`: specifies whether all possible proteins from a given genetic sequence will be translated, instead of linearly reading the sequence.
- *open_reading_frames* `bool`: specifies whether each of the three possible open reading frames for a specified sequence will be translated. The resultant proteins will be distinguished in their description for what open reading frame generated their translation.
- *filter_protein_size* `int`: specifies the peptide length below which a translated peptide will be excluded from the set of predicted proteins.
- *sense_strand_translation* `bool`: specifies whether the sense strand, complementary to the parameterzied sequence, will be translated as well.

blast_protein()

A protein sequence or a FASTA-formatted file of protein sequences is searched in through the BLAST database of the NIH for information about the protein(s):

```
protein_blast_results = cd.blast_protein(sequence = None, database = 'nr', description =
↪'Protein sequence description',  fasta_path = None, fasta_link = None,  export_name =
↪'codons-BLASTp', export_directory = None)
```

- *sequence* `str`: The protein sequence that will be searched through the BLAST database. The sequence is case-insensitive, and can even possess line numbers or column-spaces. The sequence must be < 1000 amino acids in length.
- *database* `str`: The BLAST database that will be searched for the protein sequence. Permissible options include: `nr`, `refseq_select`, `refseq_protein`, `landmark`, `swissprot`, `pataa`, `pdb`, `env_nr`, `tsa_nr`.
- *description* `str`: A description of the protein sequence that will be added to the FASTA-formatted output.
- *fasta_path* & *fasta_link* `str`: The path or URL link to a protein FASTA or multi-FASTA file that will be systematically searched.
- *export_name* & *export_directory* `str`: The name of the folder and directory to which the scraped BLAST data will be saved in iterations of `protein_blast_results.xml` XML files. The `None` values enable the code to construct a unique folder name that describes the contents and saves it to the current working directory.

Returns

- *protein_blast_results list*: The BLAST search results, which can be further investigated by the *Bio.Blast.NCBIXM* API.

blast_nucleotide()

A genetic sequence or a FASTA-formatted file of genetic sequences is searched though the BLAST database of the NIH for information about the sequence(s):

```
nucleotide_blast_results = cd.blast_nucleotide(sequence = None, database= 'nt',␣
↪description = 'Genetic sequence description', fasta_path = None, fasta_link = None,␣
↪export_name = 'codons-BLASTn', export_directory = None)
```

- *sequence* `str`: The genetic sequence, of either DNA or RNA, that will be searched through BLAST. The sequence is case-insensitive, and can even possess line numbers or column-spaces, which the code ignores. The absence of a passed sequence executes the sequence that is loaded into the `Codons` object.
- *database* `str`: The BLAST database that will be searched for the nucleotide sequence. Permissible options include: `nr`, `nt`, `refseq_select`, `refseq_rna`, `refseq_representative_genomes`, `wgs`, `refseq_genomes`, `est`, `SRA`, `TSA`, `HTGS`, `pat`, `pdb`, `RefSeq_Gene`, `gss`, `dbsts`.
- *description* `str`: A description of the genetic sequence that will be added to the FASTA-formatted output of the function.
- *fasta_path* & *fasta_link* `str`: The path or URL link to a genetic FASTA or multi-FASTA file that will be systematically searched.
- *export_name* & *export_directory* `str`: The name of the folder and directory to which the scraped BLAST data will be saved in a file: `nucleotide_blast_results.xml`. The `None` values enable the code to construct a unique folder name that describes the contents and saves it to the current working directory.

Returns

- *nucleotide_blast_results list*: The BLAST search results, which can be further investigated by the *Bio.Blast.NCBIXM* API.

export()

Any sequences from the aforementioned functions, which reside in the `Codons` object, are exported as separate files in the same folder:

```
cd.export(export_name = None, export_directory = None)
```

- *export_name* `str`: optionally specifies a name for the folder of exported content, where *None* enables the code to design a unique folder name for simulation and descriptive tags of its content.
- *export_directory* `str`: optionally specifies a path to where the folder will be exported, where *None* selects the current working directory.

Accessible content

The `Codons` object retains numerous components that are accessible to the user:

- *genes* `dict`: A dictionary of all genes in the genetic sequence, with sub-content of a) the list of all of its Codons; and b) its protein sequence and mass.
- *protein_fasta* & *gene_fasta* `str`: Assembled FASTA-formatted files for the translated proteins of a parameterized genetic sequence and for the parameterized genetic sequence, respectively.
- *transcribed_sequence* & *sequence* `str`: The transcribed genetic sequence from the `transcription()` function and the genetical sequence that is used in any of the `Codons` functions, respectively.
- *amino_acid_synonyms* `dict`: The synonyms for each amino acid, with keys of the full amino acid name.
- *codons_table* & *changed_codons* `dict`: The translation table between genetic codons and amino acid residues, which is accessed with case-insensitivity, and changed codon meanings in that table by the user, respectively.
- *missed_codons* `dict`: A collections of the codons that were parsed yet were not identified by the `codons_table`.
- *paths* & *parameters* `dict`: Collections of the paths and parameters are defined during use of `Codons`.
- *export_path* `str`: The complete export path for the `Codons` contents.
- *protein_blast_results* & *nucleotide_blast_results* `list`: The BLAST search results for searched proteins and nucleotides, respectively, which can be further investigated by the *Bio.Blast.NCBIXM* API.

2.2 Execution

Codons is executed through the following sequence of the aforementioned functions, which is exemplified in the example Notebook of our GitHub repository:

```
import codons
cd = codons.Codons(sequence = None, codons_table = 'standard', amino_acids_form = 'full_
↪name', hyphenated = None, verbose = False, printing = True)
# < Codons function(s) >
cd.export(export_name = None, export_directory = None)
```