BIOL5381 Biological Computing in Python – Script Documentation Report

Introduction

The script aims to categorize Single Nucleotide Polymorphisms (SNPs) based on whether they are present in a coding region and whether or not they cause a change in the reference amino acid they are a part of.

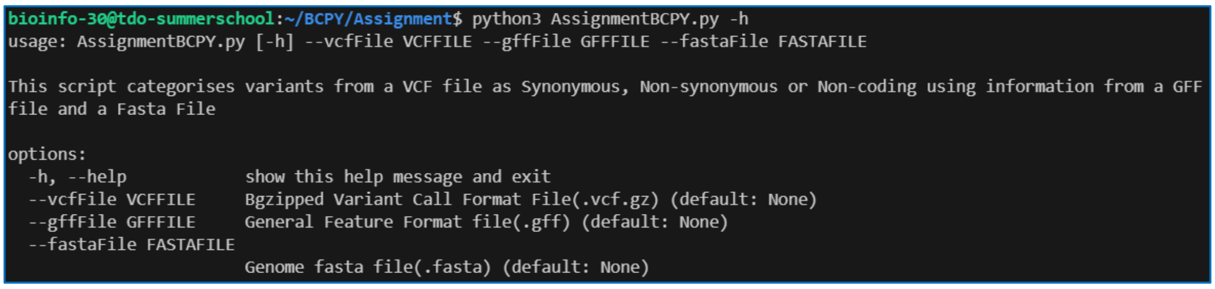
Input Files Required

The script takes in three input files

- A Variant Call Format File (VCF) containing header rows and variant information in the usual format. This file should be bgzipped. A tabix index file should also be present in the same directory but does not need to be given as an input.
- A General Feature Format File (GFF) containing annotations about Gene features such as exons and mRNA in the usual tabular format.
- A FASTA File containing the gene sequences of the organism of interest mentioned in the GFF and VCF files

Running the Script

The script is run via a command line interface implemented using the argparser module. It takes in three arguments as described above i.e "`--vcfFile`", "`--gffFile`" and "`--fastaFile`". All three arguments are compulsory and do not have default values.



```
bioinfo-30@tdo-summerschool:~/BCPY/Assignment$ python3 AssignmentBCPY.py -h
usage: AssignmentBCPY.py [-h] --vcfFile VCFFILE --gffFile GFFFILE --fastaFile FASTAFILE

This script categorises variants from a VCF file as Synonymous, Non-synonymous or Non-coding using information from a GFF
file and a Fasta File

options:
  -h, --help            show this help message and exit
  --vcfFile VCFFILE     Bgzipped Variant Call Format File(.vcf.gz) (default: None)
  --gffFile GFFFILE     General Feature Format file(.gff) (default: None)
  --fastaFile FASTAFILE
                        Genome fasta file(.fasta) (default: None)
```

Fig 1: Argparser Help Message for the script

An example command line is as follows:

```
$ python3 AssignmentBCPY.py --vcfFile assessmentData.vcf.gz --gffFile PlasmoDB-54_Pfalciparum3D7.gff --fastaFile PlasmoDB-54_Pfalciparum3D7_Genome.fasta
```

Working

After taking the command line inputs, a log file is set up to report errors, warnings and other useful information. The script then verifies that all input files exist and are in the expected formats. This is done via a user defined function - isExistingInRightFormat().

For each high quality VCF record, the script checks if the SNP is in a coding region (CDS) and then builds the transcript that CDS belongs to while tracking the position of the SNP. A transcript is also generated using the alternate variant from the VCF record. Both transcripts are translated and the amino acid to which the SNP belongs is extracted. Based on whether the reference and alternate amino acids are identical or different, the SNP is categorized as synonymous or non-synonymous. This information is presented as a tab separated table file

Other tracked information includes the number of high and low quality VCF records and the number of each category of mutation. The latter is presented as a bar graph.

Description of Output Files

The script produces three output files:

A logfile which confirms that the input files are correct, displays statistics about vcf record quality and categories of mutations, gives the location of the output files as well as any other miscellaneous warnings or errors encountered while running the script.

A bar graph of high-quality variants which are categorized into non-coding, synonymous or non-synonymous.

A tab separated table with 9 columns: CHROM, POS, REF, ALT, Type, Transcript, Protein Location, Ref AA and Alt AA. The last four columns are applicable to mutations in coding regions only.

- CHROM and POS indicate the sequence where a variant is located and the position of the SNP on that sequence respectively.
- REF and ALT refer to the Reference and Alternate(mutated) allele bases present at the given position. This information is present in the VCF file.
- The Type column gives the mutation category i.e “Non-coding”, “Synonymous” or “Non-synonymous”.
- Transcript contains the ID of the transcript(mRNA) where the mutation is located while the “Protein Location” column gives the coordinate of the amino acid containing the SNP.
- Finally, the Ref and Alt AA columns contain the amino acids coded by the reference and alternate sequences respectively, at the location where the SNP is present.

Known Issues and Possible Improvements

Building the Entire Transcript: Building and translating the entire transcript an SNP is a part of seems to be inefficient since it is only required to find the amino acid containing the SNP. However, using the translate function from the Seq module with the CDS = True parameter enabled easier checking of errors and also allowed for the automatic flagging of important Nonsense Mutations since they generate translation exceptions due to internal stop codons (A single reference sequence containing Selenocysteine also got flagged). Translating the entire transcript also removes potential issues of “reading frames” and the total execution time is less than 1 minute for the given dataset. Nevertheless, the frames are given in the GFF files and utilizing these while translating only the bases required to find the reference and alternate amino acids containing the SNP would be slightly faster.

Dictionary vs Lists: A dictionary has been used to store a single row of the tab separated table. It is more memory efficient to use a list. However, using a dict makes it easier to write and understand the code