PeptoVar USER'S MANUAL

PeptoVar (**Pept**ides **o**f **Var**iations) is a pure Python3 program for annotation of genomic variants in protein coding genes and generation of variant peptides. The main idea of PeptoVar is implementation of the potential compound effects of combining alternate alleles across multiple variant loci in sequence translation.

PeptoVar is helpful for *in-silico* prediction of new minor histocompatibility antigen (MiHa), simulation of stem sells transplantation or as peptide database generator for shotgun proteomic studies of immunopeptidome. The program can handle personalized variation data using both phased and unphased allele variants.

REQUIREMENTS

Platforms:

Linux, Mac OS

Dependencies:

python >= 3.5

pysam module >= 0.11.2.2

INSTALLATION

1) install pysam module with pip:

```
pip3 install pysam
```

2) download latest stable PeptoVar build from the project page:

https://open-projects.github.io/PeptoVar/

and unzip the archive

or clone PeptoVar using git:

```
git clone https://github.com/open-projects/PeptoVar
```

3) add resulting folder to your ``PATH`` variable

or add symbolic link for ``PeptoVar`` script to your ``bin`` folder

or use PeptoVar directly by specifying full path to the executable script

INPUT DATA AND FORMATS

- genome annotation

PeptoVar uses genome annotation data in GFF3 format (http://gmod.org/wiki/GFF3). Note that only records with CDS type are processed and grouped by Parent identifiers.

- set of genomic variations

VCF files with genetic variation data must be compressed with bgzip (block compression/decompression utility) and indexed with tabix (generic indexer for TAB-delimited genome position files). The both utilities are parts of Samtools - a suite of programs for interacting with high-throughput sequencing data (http://www.htslib.org).

USAGE

```
PeptoVar.py [-h] [-gff file.gff] [-fasta file.fasta] [-vcf filevcf.gz]
            [-samples sample_name1 [sample_name2]]
            [-tagaf TAG_AF] [-minaf THRESHOLD] [-var all | nonsyn] [-nopt]
            [-peptlen LENGTH1 [LENGTH2 ...]]
            [-tmpdir dirpath] [-outdir dirpath] [-indir dirpath]
            [-trnlist transcriptID [transcriptID ...]] [-trnfile transcriptID.txt]
            [-trnexclist transcriptID [transcriptID ...]]
            [-trnexclfile transcriptID.excl.txt] [-across] [-frame 0 | 1 | 2]
```

-h show this help message and exit

-gff GFF input file

-fasta FASTA input file (use if GFF file has no sequences)

-vcf VCF input file (requirements see in INPUT FILE FORMAT paragraph)

-samples a sample name or a pair of sample names in VCF file; for two samples (donor/recipient) only unique peptides will be represented

-tagaf allele frequency tag in VCF file (for example CAF, TOPMED, EUR_AF, SAS_AF, AMR_AF etc.); used with ‘-minaf’ argument, default=AF

-minaf allele frequency (AF) threshold; alleles with AF < THRESHOLD will be ignored (AF=0 will be set for alleles with no AF data); NOTE: ignoring or low value of -minaf can cause high memory usage and increasing the computational time

-var (all | nonsyn) save annotation of translated polymorphisms: all or only nonsynonymous

-nopt do not use optimization, i.e synonymous and non-synonymous variations will be used for sequence translation (may cause high CPU load and memory usage)
-peptlen lengths of peptides (0 - full-length proteins)
-tmpdir temporary directory
-outdir output directory (will be created if not exists, default: ./output)
-indir input directory for files *.vcf.gz, *.vcf.gz.tbi, *.gff and *.fasta (if no sequences in GFF file); the files MUST have the same name for each locus (chromosome)
-trnlist transcriptID list for processing
-trnfile one column text file with the transcriptID set for processing
-trnexclist transcriptID list to EXCLUDE from processing
-trnexclfile one column text file with the transcriptID set to EXCLUDE from processing
-across translate across stop codons (peptides with stop codons will not be presented in the output)
-frame frame of translation: 0 – frame of the reference annotation; 1,2 – alternative frames (default is '0')

MAIN APPLICATIONS AND EXAMPLES

1. To get annotation of the genome variations in protein coding genes (**annotation mode**):

```
PeptoVar.py –var used –gff ./testdata/test.gff –vcf ./testdata/test.vcf.gz
```
or
```
PeptoVar.py –var used –indir ./testdata
```

The mode can be combined with any other.

2. To generate peptides for all combinations of genome variations in a population (**population mode**):

```
PeptoVar.py –peptlen 8 9 10 –gff ./testdata/test.gff –vcf ./testdata/test.vcf.gz
```
or (combined with annotation mode)
```
PeptoVar.py –peptlen 8 9 10 –var used –indir ./testdata
```

3. To make personalized peptidomes for selected samples (**sample mode**):

```
PeptoVar.py –samples SAMPLE01 –peptlen 9 –gff ./testdata/test.gff \
–vcf ./testdata/test.vcf.gz
```
or

```
PeptoVar.py –samples SAMPLE01 –peptlen 9 –indir ./testdata
```

4. To get unique peptides for a sample in a pair of samples (**transplantation mode**). Note that unique peptides are calculated for each sample, i. e. for transplantation in both directions:

```
PeptoVar.py –samples SAMPLE01 SAMPLE02 –peptlen 9 –gff ./testdata/test.gff \
–vcf ./testdata/test.vcf.gz
```

or

```
PeptoVar.py –samples SAMPLE01 SAMPLE02 –peptlen 9 –indir ./testdata
```

“sample” and “transplantation” modes are adapted to multi-sample VCF files (like the 1000 Genome project VCF files)

OUTPUT FILE FORMATS

PeptoVar creates four files in CSV format (TAB-delimited) in the output directory: the file for peptides (if `-peptlen` option has the value more than ‘0’), the file for proteins (if `-peptlen` option has ‘0’ value), the file for polymorphisms (if `-var` option is utilized) and the file for warnings.

Data fields in the file for peptides (sample_name.pept.csv):

chrom – chromosome number (or locus name)

transcript_id – transcript ID

sample – sample name

sample_allele1 – presence in allele 1 (‘0’ - not in allele 1; ‘1’ - in allele 1)

sample_allele2 – presence in allele 2 (‘0’ - not in allele 2; ‘1’ - in allele 2)

beg – start genome position of the peptide*

end – end genome position of the peptide*

upstream_fshifts – frame shifts in upstream of the peptide which induced the frame translation (if several combinations of frame shifts can induce the current frame the full set of the combinations will be presented with ‘|’ delimiter)

variations(positions_in_matrix) - polymorphisms inside the peptide and their relative nucleotide positions

peptide – peptide sequence

matrix – genome sequence of translation to the peptide

* - if peptide translation starts and/or ends on a genome insertion the peptide genome positions will be calculated as *genome_position = leftmost_translated_reference_position + 0.5* and formated as "genome_pos+insertion_position(variation_id:variation_allele_sequence(alt|ref))"

Data fields in the file for proteins (sample_name.prot.csv):

chrom – chromosome number (or locus name)

transcript_id – transcript ID

sample – sample name

sample_allele1 – presence in allele 1 ('0' - not in allele 1; '1' - in allele 1)

sample_allele2 – presence in allele 2 ('0' - not in allele 2; '1' - in allele 2)

beg – start genome position of the protein

end – end genome position of the protein

variations(positions_in_matrix) - polymorphisms inside the protein and their nucleotide positions

peptide – protein sequence

matrix – genome sequence of translation to the protein

Data fields in the file for polymorphisms (variations.csv):

transcript_id – transcript ID

variation_id – polymorphism ID

beg – start genome position of the polymorphism

end – end genome position of the polymorphism

allele_id – allele ID (polymorphism_ID:polymorphism_sequence=allele_frequency(ref|alt), where '=allele_frequency' is reported only for defined values in VCF files; ref - reference sequence, alt - alternative sequence)

sample – sample name

sample_allele1 – presence in allele 1 ('0' - not in allele 1; '1' - in allele 1)

sample_allele2 – presence in allele 2 ('0' - not in allele 2; '1' - in allele 2)

synonymous – '1' if the polymorphism is synonymous, else '0'

upstream_fshifts – frame shifts in upstream of the polymorphism which induced the frame translation (if several combinations of frame shifts can induce the current frame the full set of the combinations will be presented with '|' delimiter)

prefix_alleles – polymorphisms in the polymorphism prefix

prefix – sequence of the polymorphism prefix (the nucleotide sequence from the start of current codon to the start of polymorphism)

allele – sequence of the polymorphism

suffix – sequence of the polymorphism suffix (the nucleotide sequence from the end of the polymorphism to the end of current codon)
suffix_alleles – polymorphisms in the polymorphism suffix
translation – translation of the polymorphism with the suffix and the prefix

NOTE*: ID of synonymous polymorphism alleles are enclosed in square brackets*

File with warnings (warnings.csv):
PeptoVar inspect records in the VCF and the GFF files for collisions.
The most important ones:
- multiple polymorphisms with the same ID
- multiple polymorphisms with the same genome positions but different ID
- intersection of a polymorphism and an exon border (in this case it is unknown if the polymorphism changes the site of splicing)
- no data for the sample in the VCF file
- no genome sequence in the exon position range

LICENSE