ProteoGenomics Analysis ToolKit Documentation

Release latest

Aug 19, 2021

Contents

Welcome to the ProteoGenomics Analysis ToolKit (PGATK), a framework for proteogenomics analysis. It provides a set of tools and bioinformatics pipelines to perform proteogenomics analysis using proteomics and RNA-Seq data.

CHAPTER 1

Contents

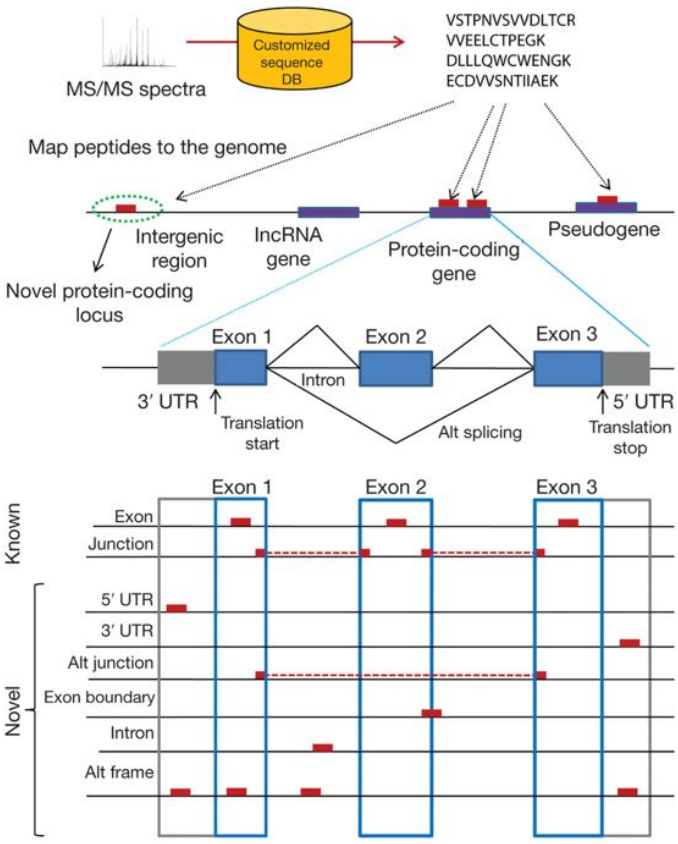
1.1 Introduction

Individual Tools Documentation

It can make your life easier if you want to explore individual tools:

- *PepGenome: Mapping Peptidoforms to Genome Coordinates*
- *PepBedR: R package to analyze peptidoforms features in Bed Files*

Proteogenomics is a field of biological research that utilizes a combination of proteomics, genomics, and transcriptomics to aid in the discovery and identification/quantification of peptides and proteins. Proteogenomics is used to identify new peptides by comparing MS/MS spectra against a protein database that has been derived from genomic and transcriptomics information.



In this approach, customized protein sequence databases generated using genomic and transcriptomic information are used to help identify novel peptides (not present in reference protein sequence databases) from mass spectrometry–based proteomic data; in turn, the proteomic data can be used to provide protein-level evidence of gene expression and to help refine gene models.

Proteogenomics can be also seen as the way to present proteomics evidences in to genomics context.

Note: You can read a more about the topic here: https://www.nature.com/articles/nmeth.3144

1.2 PGATK Tools

1.2.1 PepGenome: Mapping Peptidoforms to Genome Coordinates

In proteogenomic analyses it is essential to know the loci (genome position) giving rise to **peptides** in order to improve genomic annotation and the functional characterization of protein products in their biological context. With next-generation sequencing of DNA and RNA for each sample studied by proteomic mass spectrometry integration and visualisation in a common coordinate system, i.e. the genome, is vital for systems biology. To facilitate this type of integration not only the genomic locations of modified peptides but specifically the genomic loci of associated with these modifications is required.

Note: The **PepGenome** tool quickly and efficiently identify genomic loci of peptides and post-translational modifications and couple these mappings with associated quantitative values. Using reference gene annotation (GTF files) and an associated transcript translations (Protein fasta files) our tool identifies the genomic loci of peptides given as

input and generates output in different formats borrowed from genomics and transcriptomics which can be loaded in various genome browsers such as UCSC Genome Browser, Ensembl Genome Browser.

Input format (Tab delimited)

The input format required by PepGenome is a tab delimited file with four columns. It can contains the following extensions `.tsv`, `.pogo` or `.txt`.

Column	Column header	Description
1	Sample	Name of sample or experiment
2	Peptide	Peptide sequence *
3	PSMs	Number (PSMs)
4	Quant	Quantitative value in the given sample

Hint: *Peptide sequence with PSI-MS modification names in round brackets following the modified amino acid, e.g. PEPT(Phopsho)IDE for a phosphorylated threonine.

Note: In addition the tool support mzTab and mzIdentML File format input.

Output formats

BED Files

This format contains the genomic loci for peptides, the exon-structure, the peptide sequence, as well as a colour code for uniqueness of peptides within the genome. Read here *BED Format*

Additional Files

- GTF: This output format contains besides the genomic loci the annotated information for the genes giving rise to each peptide sequence including status and biotype. For each mapped peptide the sample, number of peptide-spectrum matches and associated quantitative value as tags.
- GCT: In this format the peptide sequences are combines with the Ensembl gene identifier. It contains the genomic loci for each peptide as well as the quantitative values for each peptide in different samples as a matrix.

Usage

Required arguments:

- `-fasta`: Filepath for file containing protein sequences in FASTA format. (e.g. `-fasta gencode.v25.pc_translations.fa`)
- `-gtf`: Gene annotation with coding sequences (CDS) in GTF format. (e.g. `-gtf gencode.v25.annotation.gtf`)
- `-in`: Path to single input file or comma separated list of paths to input files containing peptides to be mapped with associated number of peptide to spectrum matches, sample name and quantitative value (see input file format). (e.g. `-in file.tsv`)

How to easily run the tool (**e.g. Human**):

```
$ java -jar -Xmx5G PepGenome-{version}-bin.jar -gtf gencode-{version}.gtf
  -fasta gencode-{version}-translations.fa -in file.tsv
```

Note: the tool can be download from PepGenome Releases

Optional arguments

- `-format`: Set output format _GTF_, _GCT_, _BED_, _PTMBED_ or _ALL_. Comma separated combination possible. `Default = ALL`
- `-merge`: Set TRUE/FALSE to merge output of multiple input files (output will be named after last input file *_merged). `Default = FALSE`
- `-source`: Set TRUE/FALSE to merge output of multiple input files (output will be named after last input file *_merged). `Default = FALSE`
- `-mm`: Number of mismatches allowed in mapping (0, 1 or 2). `DEFAULT = 0`
- `-mmmode`: Set TRUE/FALSE to restrict number of mismatch in kmer to 1. `DEFAULT = FALSE`.
- `-genome`: Filepath for the fine containing genome sequences in Ensembl FASTA format. Used to identify chromosome names and order and differenciate between chromosomes and scaffolds. If not set chromosome names are extracted from the GTF file without differenciation between chromosomes and scaffolds. (e.g. ‘‘ -genome Homo_sapiens.GRCh38.89.dna.primary_assembly.fa‘‘)
- `-chr`: Export chr prefix Allowed 0, 1. (e.g. `-chr 1`) `DEFAULT = 0`

1.2.2 Pypgatk: Python Tools for ProteoGenomics

The Pypgatk framework and library provides a set of tools to perform proteogenomics analysis. In order to execute a task in `pypgatk` the user should use a `COMMAND` to perform the specific task and specify the task arguments:

```
$: pypgatk_cli.py -h
   Usage: pypgatk_cli.py [OPTIONS] COMMAND [ARGS]...

   This is the main tool that gives access to all commands and options provided by␣
↪the pypgatk_cli

   Options:
      -h, --help  Show this message and exit.

   Commands:
     ensembl-downloader       Command to download the ensembl information
     cbioportal-downloader    Command to download the the cbioportal studies
     cosmic-downloader        Command to download the cosmic mutation database
     dnaseq-to-proteindb      Command to translate sequences generated from RNA-seq␣
↪and DNA sequences
     vcf-to-proteindb         Command to translate genomic variatns to protein␣
↪sequences
     cbioportal-to-proteindb  Command to translate cbioportal mutation data into␣
↪proteindb
     cosmic-to-proteindb      Command to translate Cosmic mutation data into proteindb
```

(continues on next page)

(continued from previous page)

```
    generate-decoy          Command to generate decoy database from a proteindb
    ensembl-check               Command to fix protein sequences to only contain␣
↪amino acid sequences
```

Installation

Clone the source code for pypgatk from source:

```
git clone https://github.com/bigbio/py-pgatk.git
```

`pypgatk` depends on several `Python3` packages that are listed in `requirements.txt`, once in the downloaded directory install the dependencies using `pip`:

```
pip install -r requirements.txt
```

Install the `pypgatk` package from source:

```
python setup.py install
```

Data Downloader Tools

The Data downloader is a set of COMMANDs to download data from different Genomics data providers including ENSEMBL, COSMIC and cBioPortal.

Downloading ENSEMBL Data

Downloading data from ENSEMBL can be done using the command *ensembl_downloader*. The current tool enables downloading the following files for any taxonomy that is available ENSEMBL:

- GTF
- Protein Sequence (FASTA)
- CDS (FASTA)
- CDNA sequences (FASTA)
- Non-coding RNA sequences (FASTA)
- Nucleotide Variation (VCF)
- Genome assembly DNA sequences (FASTA)

Command Options

```
$: python pypgatk_cli.py ensembl-downloader -h
   Usage: pypgatk_cli.py ensembl-downloader [OPTIONS]

   This tool enables to download from ENSEMBL ftp the FASTA, GTF and VCF files

    Required parameters::
     -c, --config_file TEXT          Configuration file for the ensembl data␣
↪downloader pipeline
```

(continues on next page)

(continued from previous page)

```
    -o, --output_directory TEXT     Output directory for the peptide databases

  Optional parameters:
    -l, --list_taxonomies TEXT      List the available species from Ensembl, users can find the desired taxonomy identifier from this list.
    -fp, --folder_prefix_release    TEXT Output folder prefix to download the data
    -t, --taxonomy TEXT             Taxonomy identifiers (comma separated) that will be use to download the data from Ensembl
            -sv, --skip_vcf                 Skip the vcf file during the download
    -sg, --skip_gtf                 Skip the gtf file during the download
    -sp, --skip_protein             Skip the protein fasta file during download
    -sc, --skip_cds                 Skip the CDS file download
            -sn, --skip_ncrna               Skip the ncRNA file download
    -sdn, --skip_cdna               Skip the cDNA file download
    -sd, --skip_dna                 Skip the DNA file download
    -h, --help                      Show this message and exit.
```

Examples

- List all species without downloading any data:

```
python pypgatk_cli.py ensembl-downloader -l -sv -sg -sp -sc -sd -sn
```

- Download all files except cDNA for Tureky (species id=9103, note that th species id cab be obtained from the list above):

```
python pypgatk_cli.py ensembl-downloader -t 9103 -sd -o ensembl_files
```

- *[To be implemented]* Download CDS file for Humans (species id=9606) from release 94 and genome assembly GRCh37

```
python pypgatk_cli.py ensembl-downloader -t 9606 -sv -sg -sp -sd -sn -o ensembl_files --release 94 --assembly GRCh37
```

Note: By default the command `ensembl-downloader` downloads all datasets for all species from the latest ENSEMBL release. To limit the download to a particular species specify the species identifier using the `-t` option. To list all available species run the command with `-l (--list_taxonomies)` option.

Note: Any of the file types can be skipped using the corresponding option. For example, to avoid downloading the protein sequence fasta file, use the argument `--skip_protein`. Also, note that not all file types exists for all species so obviously the downloaded files depends on availabiliy of the dataset in ENSEMBL.

Hint: a VCF file per chromosome is downloaded for homo sapiens due to the large file size they have been distributed this way by ENSEMBL. For other species, a single VCF including all chromosomes is downloaded.

Downloading COSMIC Data.

Downloading mutation data from COSMIC is performed using the COMMAND `cosmic-downloader`. The current COMMAND allows users to download the following files:

- Cosmic mutation file (CosmicMutantExport)
- Cosmic all genes (All_COSMIC_Genes)

Command Options

```
$: python pypgatk_cli.py cosmic-downloader -h
   Usage: pypgatk_cli.py cosmic-downloader [OPTIONS]

   Required parameters:
     -u, --username TEXT          Username for cosmic database -- please if you dont␣
↪have one register here (https://cancer.sanger.ac.uk/cosmic/register)
     -p, --password TEXT          Password for cosmic database -- please if you dont␣
↪have one register here (https://cancer.sanger.ac.uk/cosmic/register)

        Optional parameters:
     -c, --config_file TEXT       Configuration file for the ensembl data downloader␣
↪pipeline
     -o, --output_directory TEXT  Output directory for the peptide databases
     -h, --help                   Show this message and exit.
```

Note: In order to be able to download COSMIC data, the user should provide a user and password. Please first register in COSMIC database (https://cancer.sanger.ac.uk/cosmic/register).

Examples

- Downlaod `CosmicMutantExport.tsv.gz` and `All_COSMIC_Genes.fasta.gz`:

```
python pypgatk_cli.py cosmic-downloader -u userName -p passWord -c config/cosmic_
↪config.yaml -o cosmic_files
```

Downloading cBioPortal Data.

Downloading mutation data from cBioPortal is performed using the command `cbioportal-downloader`. cBioPortal stores mutation data from multiple studies (https://www.cbioportal.org/datasets). Each dataset in cBioPortal has an associated study_id.

Command Options

```
$: python3.7 pypgatk_cli.py cbioportal-downloader -h
   Usage: pypgatk_cli.py cbioportal-downloader [OPTIONS]

   Parameters:
     -c, --config_file TEXT Configuration file for the ensembl data downloader␣
↪pipeline
     -o, --output_directory TEXT  Output directory for the peptide databases
     -l, --list_studies           Print the list of all the studies in cBioPortal␣
↪(https://www.cbioportal.org)
     -d, --download_study TEXT    Download a specific Study from cBioPortal -- (all␣
↪to download all studies)
     -h, --help                   Show this message and exit.
```

Note: The argument `-l` (`--list_studies`) allows the user to list all the studies stored in cBioPortal. The `-d` (`--download_study`) argument can be used to obtain mutation data from a particular study.

Examples

- Download data for study ID blca_mskcc_solit_2014:

```
python pypgatk_cli.py cbioportal-downloader -d blca_mskcc_solit_2014 -o cbiportal_
↪files
```

- Download data for all studies in cBioPortal:

```
python pypgatk_cli.py cbioportal-downloader -d all -o cbioportal_files
```

If you face issues downloading all studies from cBioPortal using the `cbioportal-downloader`, please download the studies from the data hub through `git-lfs` which is used to download large files from gitHub repositories, see installation instructions:.

Following instructions given on the datahub repository, download the entire list of datasets using:

```
git clone https://github.com/cBioPortal/datahub.git
cd datahub
git lfs install --local --skip-smudge
git lfs pull -I public --include "data_clinical_sample.txt"
git lfs pull -I public --include "data_mutations_mskcc.txt"
```

Generate Protein Databases

The **Pypgatk** framework provides a set of tools (COMMAND) to generate protein databaseas in `FASTA` format from DNA sequences, variants, and mutations. In order to perform this task, we have implemented multiple commands depending on data type provided by the user and the public data providers (cBioPortal, COSMIC and ENSEMBL).

Cosmic Mutations to Protein Sequences

COSMIC the Catalogue of **Human** Somatic Mutations in Cancer – is the world's largest source of expert manually curated somatic mutation information relating to human cancers. The command `cosmic-to-proteindb` converts the cosmic somatic mutations file into a protein sequence database file.

Command Options

```
$: python pypgatk_cli.py cosmic-to-proteindb -h
   Usage: pypgatk_cli.py cosmic-to-proteindb [OPTIONS]

   Required parameters:
     -in, --input_mutation TEXT   Cosmic Mutation data file
     -fa, --input_genes TEXT      All Cosmic genes
     -out, --output_db TEXT       Protein database including all the mutations

   Optional parameters:
     -c, --config_file TEXT       Configuration file for the cosmic data pipelines
     -f, --filter_column          Column name to use for filtering or splitting␣
↪mutations by, default value is ``Primary site``
```

(continues on next page)

(continued from previous page)

```
    -a, --accepted_values        Only consider mutations from records that belong to these groups as specified by ``-filter_column`` option, by default mutations from all groups are considered (default ``all``)
    -s,     --split_by_filter_column Generate a proteinDB output file for each group in the mutations file (affected by ``--filter_column``) (default ``False``)
    -h, --help                   Show this message and exit.
```

The file input of the tool `-in` (`--input_mutation`) is the cosmic mutation data file. The genes file `-fa` (`--input_genes`) contains the original CDS sequence for all genes used by the COSMIC team to annotate the mutations. *Use cosmic-downloader* to obtain the input files from COSMIC.

The output of the tool is a protein fasta file and is written in the following path *-out* (`--output_db`)

Examples

- Generate cancer-type specific protein databases. For each cancer type in COSMIC generate a protein database based on the Primary site given in the mutations file:

```
python pypgatk_cli.py cosmic-to-proteindb -in CosmicMutantExport.tsv -fa All_COSMIC_Genes.fasta -out cosmic_proteinDB.fa --split_by_filter_column
```

- Generate cell-line specific protein databases. For each cell line in COSMIC cell lines generate a protein database based on the Sample name given in the mutations file:

```
python pypgatk_cli.py cosmic-to-proteindb -in CosmicCLP_MutantExport.tsv -fa All_CellLines_Genes.fasta -out cosmicCLP_proteinDB.fa --split_by_filter_column --filter_column 'Sample name'
```

cBioPortal Mutations to Protein Sequences

The cBioPortal for Cancer Genomics provides visualization, analysis and download of large-scale cancer genomics data sets. The available datasets can be viewed in this web page (https://www.cbioportal.org/datasets). The command `cbioportal-to-proteindb` converts the bcioportal mutations file into a protein sequence database file.

Command Options

```
$: python pypgatk_cli.py cbioportal-to-proteindb -h
   Usage: pypgatk_cli.py cbioportal-to-proteindb [OPTIONS]

    Required parameters:
     -c, --config_file TEXT           Configuration for cBioportal
     -in, --input_mutation TEXT       Cbioportal mutation file
     -fa, --input_cds TEXT            CDS genes from ENSEMBL database
     -out, --output_db TEXT           Protein database including the mutations

    Optional parameters:
     -f, --filter_column TEXT         Column in the VCF file to be used for filtering or splitting mutations
     -a, --accepted_values TEXT       Limit mutations to groups (values) (tissue type, sample name, etc) considered for generating proteinDBs, by default mutations from all records are considered
     -s, --split_by_filter_column     Use this flag to generate a proteinDB per group as specified in the filter_column, default is False
```

(continues on next page)

(continued from previous page)

```
    -cl, --clinical_sample_file TEXT  Clinical sample file that contains the cancery␣
↪type per sample identifier (required when ``-t`` or ``-s`` is given).
    -h, --help                        Show this message and exit.
```

Note: The clinical sample file for each mutation file can be found under the same directory as the mutation file downloaded from cBioportal (It should have at least two columns named: Cancer Type and Sample Identifier). The file is only needed if generating tissue type databases is desired (that is when -s or -a is given).

The file input of the tool `-in` (`--input_mutation`) is the cbioportal mutation data file. An example is given in *cbioportal-downloader* showing how to obtain the mutations file for a particular study. The CDS sequence for all genes input file `-fa` (`--input_genes`) can be obtained using the ENSEMBL CDS files, see *this example*. The output of the tool is a protein fasta file and it is written in the following path `-out` (`--output_db`)

Note: The cBioportal mutations are aligned to the hg19 assembly, make sure that the correct genome assembly is selected for the download.

Examples

- translate mutations from `Bladder` samples in studyID: `blca_mskcc_solit_2014` (*use cbioportal-downloader* to download the study, then extract the content of the downloaded file):

```
python pypgatk_cli.py cbioportal-to-proteindb --config_file config/cbioportal_
↪config.yaml --input_cds human_hg19_cds.fa  --input_mutation data_mutations_
↪mskcc.txt --clinical_sample_file data_clinical_sample.txt --output_db bladder_
↪proteindb.fa
```

Variants (VCF) to Protein Sequences

Variant Calling Format (VCFv4.1) is a text file representing genomic variants.

The `vcf_to_proteindb` COMMAND takes a VCF file and a GTF (Gene annotations) file to translates the genomic variants in the VCF that affect protein-coding transcripts.

Command Options

```
$: python pypgatk_cli.py vcf-to-proteindb -h
  Usage: pypgatk_cli.py vcf-to-proteindb [OPTIONS]

  Required parameters:
    -c, --config_file TEXT      Configuration for VCF conversion parameters
    -v, --vcf                             VCF file containing the genomic variants
    -g, --gene_annotations_gtf  Gene models in the GTF format that will be used to␣
↪extract protein-coding transcripts
    -f, --input_fasta               Fasta sequences for the transripts in the GTF␣
↪file used to annotated the VCF
    -o, --output_proteindb      Output file to write the resulting variant protein␣
↪sequences

  Options:
    --translation_table INTEGER     Translation table (Default 1). Please see
↪<https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi> for identifiers of␣
↪translation tables.
```

(continues on next page)

(continued from previous page)

```
    --mito_translation_table INTEGER        Mito_trans_table (default 2), also from
↪<https://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi>
    --var_prefix TEXT       String to add as prefix for the variant peptides
    --report_ref_seq        In addition to variant peptides, also report the␣
↪reference peptide from the transcripts overlapping the variant
    --annotation_field_name TEXT    Annotation Field name found in the INFO column,␣
↪e.g CSQ or vep, set to empty if the VCF is not annotated (default is CSQ)
    --af_field TEXT Field name in the VCF INFO column that shows the variant allele␣
↪frequency (VAF, default is none).
    --af_threshold FLOAT      Minium allele frequency threshold for considering the␣
↪variants
    --transcript_index INTEGER      Index of transcript ID in the annotated columns␣
↪in the VCF INFO field that is when the VCF file is alerady annotated, affected by --
↪annotation_field_name (separated by |) (default is 3)
    --consequence_index INTEGER     Index of consequence in the annotated columns in␣
↪the VCF INFO field that is when the VCF file is alerady annotated, affected by --
↪annotation_field_name (separated by |) (default is 1)
    --include_consequences TEXT     Consider variants that have one of these␣
↪consequences, affected by --annotation_field_name  (default is all) (for the list␣
↪of consequences see: https://www.ensembl.org/info/genome/variation/prediction/
↪predicted_data.html.
    --exclude_consequences TEXT     Variants with these consequences will not be␣
↪considered for translation, affected by --annotation_field_name  (default:␣
↪downstream_gene_variant, upstream_gene_variant, intergenic_variant, intron_variant,␣
↪synonymous_variant)
    --skip_including_all_cds        By default any affected transcript that has a␣
↪defined CDS will be translated, this option disables this features instead it only␣
↪depends on the specified biotypes
    --ignore_filters        Enabling this option causes all variants to be parsed.␣
↪By default only variants that have not failed any filters will be processed (FILTER␣
↪field is PASS, None, .) or if the filters are subset of the accepted_filters␣
↪(default is False)
    --accepted_filters TEXT Accepted filters for variant parsing
    -h, --helP              Show this message and exit.
```

The file input of the tool `--vcf` is a VCF file that can be provided by the user or obtained from ENSEMBL using *ensembl_downloader*, see *an example here*. The `gene_annotations_gtf` file can also be obtained with the *ensembl_downloader*.

The `--input_fasta` file contains the `CDS` and DNA sequences for all genes present in the GTF file. This file can be generated from the GTF file using the gffread tool as follows:

```
$: gffread -F -w input_fasta.fa -g genome.fa gene_annotations_gtf
```

The output of the tool is a protein fasta file and is written in the following path `--output_proteindb`.

Examples

- Translate human *missense* variants from ENSEMBL VCFs that have a minimum *AF 5%*:

```
python pypgatk_cli.py vcf-to-proteindb
    --vcf homo_sapiens_incl_consequences.vcf
    --input_fasta transcripts.fa
    --gene_annotations_gtf genes.gtf
    --include_consequences missense_variant
    --af_field MAF
    --af_threshold 0.05
    --output_proteindb var_peptides.fa
```

Note:

- By default vcf-to-proteindb considers transcript that have a coding sequence that includes all protein_coding genes.
- by default all consequences are accepted except those given with `--exclude_biotypes`. See the list consequences of consequences generated by VEP: https://www.ensembl.org/info/genome/variation/prediction/predicted_data.html

- Translate human *missense* variants or *inframe_insertion* from gnoMAD VCFs that have a minmum 1% allele frquency in control samples:

```
python pypgatk_cli.py vcf-to-proteindb
   --vcf gnmad_genome.vcf
   --input_fasta gencode.fa
   --gene_annotations_gtf gencode.gtf
   --include_consequences missense_variant,frameshift_insert
   --annotation_field_name vep
   --af_threshold 0.01
   --af_field control_af
   --transcript_index 6
```

Hint:

- `vcf-to-proteindb` considers transcript that have a coding sequence which includes all *protein_coding* transcripts.
- The provided VCF file has some specific properties: the annotation field is specified with the string *vep* hence the `--annotation_field_name parameter`, the transcriptat the sixth position in the annotation field, and since gnomAD collects variants from many sources it provides allele frequencies across many many sub-populations and sub-groups, in this case the goal is to use only variants that are common within control samples therefroe the `--af_field` is set to `control_af`.
- Since gnomAD uses GENCODE gene annotations for annotation the variants we need to change the default `biotype_str` from *transcript_biotype* to *transcript_type* (as written in the GTF file).

Note: As shown in the two examples above, when ENSEMBL data is used, the default options should work. However, for using other data sources such as variants from gnomAD, GTF from GENOCODE and others one or more of the following parameters need to be changed:

–af_field (from the VCF INFO field)

—annotation_field_name (from the VCF INFO field)

–transcript_index (from the annotation field in the VCF INFO field)

—consequence_index (from the annotation field in the VCF INFO field)

- Translate human variants from a custom VCF that is obtained from sequencing of a sample:

```
python pypgatk_cli.py vcf-to-proteindb
   --vcf sample.vcf
   --input_fasta transcripts.fa
   --gene_annotations_gtf genes.gtf
```

(continues on next page)

(continued from previous page)

```
  --annotation_field_name ''
  --output_proteindb var_peptides.fa
```

Transcripts (DNA) to Protein Sequences

DNA sequences given in a fasta format can be translated using the `dnaseq-to-proteindb` tool. This tool allows for translation of all kinds of transcripts (coding and noncoding) by specifying the desired biotypes. The most suited `--input_fasta` file can be generated from a given GTF file using the `gffread` commad as follows:

```
$: gffread -F -w transcript_sequences.fa -g genome.fa gene_annotations_gtf
```

The fasta file that is generated from the GTF file would contain DNA sequences for all transcripts regardless of their biotypes. Also, it specifies the CDS positions for the protein coding transcripts. The `dnaseq-to-proteindb` command recognizes the features such as biotype and expression values in the fasta header that are taken from the GTF INFO filed (if available). However, it is not required to have those information in the fasta header but their presence enables the user to filter by biotype and expression values during the translation step.

Command Options

```
$: python pypgatk.py dnaseq-to-proteindb -h
  Usage: pypgatk.py dnaseq-to-proteindb [OPTIONS]

  Required parameters:
    -c, --config_file TEXT      Configuration for VCF conversion parameters
    --input_fasta         Fasta sequences for the transripts in the GTF file used to annotated the VCF
    --output_proteindb          Output file to write the resulting variant protein sequences

  Optional parameters:
      --translation_table INTEGER    Translation Table (default 1)
      --num_orfs INTEGER             Number of ORFs (default 0)
      --num_orfs_complement INTEGER  Number of ORFs from the reverse side (default 0)
      --skip_including_all_cds       By default any transcript that has a defined CDS will be translated, this option disables this features instead it only depends on the biotypes
      --include_biotypes TEXT        Translate sequences with the spcified biotypes. Multiple biotypes can be given separated by comma. To translate all sequences in the input_fasta file set this option to ``all`` (default protein coding genes).
      --exclude_biotypes TEXT        Skip sequences with unwanted biotypes (affected by --include_biotypes) (default None).
      --biotype_str TEXT             String used to identify gene/transcript biotype in the fasta file (default transcript_biotype).
      --expression_str TEXT          String to be used for extracting expression value (TPM, FPKM, etc) (default None).
      --expression_thresh FLOAT      Threshold used to filter transcripts based on their expression values (default 5, affected by --expression_str)
      --var_prefix TEXT              Prefix to be added to fasta headers (default none)
      -h, --help                     Show this message and exit
```

Examples

- Generate the canonical protein database, i.e. translate all *protein_coding* transcripts:

```
python pypgatk.py dnaseq-to-proteindb
    --config_file config/ensembl_config.yaml
    --input_fasta testdata/transcript_sequences.fa
    --output_proteindb testdata/proteindb_from_CDSs_DNAseq.fa
```

- Generate a protein database from lincRNA and canonical proteins:

```
python pypgatk.py dnaseq-to-proteindb
    --config_file config/ensembl_config.yaml
    --input_fasta testdata/transcript_sequences.fa
    --output_proteindb testdata/proteindb_from_lincRNA_canonical_sequences.fa
    --var_prefix lincRNA_
    --include_biotypes lincRNA
```

- Generate a protein database from processed pseudogene:

```
python pypgatk.py dnaseq-to-proteindb
    --config_file config/ensembl_config.yaml
    --input_fasta testdata/transcript_sequences.fa
    --output_proteindb testdata/proteindb_from_processed_pseudogene.fa
    --var_prefix pseudogene_
    --include_biotypes processed_pseudogene,transcribed_processed_pseudogene,
↪translated_processed_pseudogene
    --skip_including_all_cds
```

- Generate alternative ORFs from canonical sequences:

```
python pypgatk.py dnaseq-to-proteindb
    --config_file config/ensembl_config.yaml
    --input_fasta testdata/transcript_sequences.fa
    --output_proteindb testdata/proteindb_from_altORFs.fa
    --var_prefix altorf_
    --include_biotypes altORFs
    --skip_including_all_cds
```

- Generate protein sequences (six-frame translation) from a Genome assembly:

```
python pypgatk.py dnaseq-to-proteindb
    --config_file config/ensembl_config.yaml
    --input_fasta testdata/genome.fa
    --output_proteindb testdata/proteindb_genome.fa
    --biotype_str ''
    --num_orfs 3
    --num_orfs_complement 3
```

Generate Decoy Database

`generate-decoy` command enables generation of decoy databases for any given protein sequence database. Decoy databases are need to evaluate significance of spectra-sequence matching scores in proteomics mass spectrometry experiments.

DecoyPYrat is integrated into `py-pgatk` as the standard method for generating decoy sequences. In addition to reversing the target sequences, the tool replaces the cleavage with preceding amino acids. Also, it checks for the presence of the reversed sequence in the target sequences and if found, *DecoyPYrat* shuffles the sequences to avoid target-decoy sequence matches. For more information please read the *DecoyPYrat* manual available at: https://www.sanger.ac.uk/science/tools/decoypyrat.

Command Options

```
$: python pypgatk.py dnaseq-to-proteindb -h
   Usage: pypgatk.py dnaseq-to-proteindb [OPTIONS]

   Required parameters:
     -c, --config_file TEXT          Configuration file for the protein database␣
↪decoy generation
     -o, --output TEXT               Output file for decoy database
     -i, --input TEXT                FASTA file of target protein sequences for
                                     which to create decoys (*.fasta|*.fa)
   Optional parameters:
     -s, --cleavage_sites TEXT       A list of amino acids at which to cleave
                                     during digestion. Default = KR
     -a, --anti_cleavage_sites TEXT  A list of amino acids at which not to cleave
                                     if following cleavage site ie. Proline.
                                     Default = none
     -p, --cleavage_position TEXT    Set cleavage to be c or n terminal of
                                     specified cleavage sites. Options [c, n],
                                     Default = c
     -l, --min_peptide_length INTEGER
                                     Set minimum length of peptides to compare
                                     between target and decoy. Default = 5
     -n, --max_iterations INTEGER    Set maximum number of times to shuffle a
                                     peptide to make it non-target before
                                     failing. Default=100
     -x, --do_not_shuffle TEXT       Turn OFF shuffling of decoy peptides that
                                     are in the target database. Default=false
     -w, --do_not_switch TEXT        Turn OFF switching of cleavage site with
                                     preceding amino acid. Default=false
     -d, --decoy_prefix TEXT         Set accession prefix for decoy proteins in
                                     output. Default=DECOY_
     -t, --temp_file TEXT            Set temporary file to write decoys prior to
                                     shuffling. Default=protein-decoy.fa
     -b, --no_isobaric TEXT          Do not make decoy peptides isobaric.
                                     Default=false
     -m, --memory_save TEXT          Slower but uses less memory (does not store
                                     decoy peptide list). Default=false
     -h, --help                      Show this message and exit.
```

Examples

- Generate decoy sequences for `proteindb_from_lincRNA_canonical_sequences.fa` that was generate using *dnaseq-to-proteindb*:

```
python pypgatk_cli.py generate-decoy -c config/protein_decoy.yaml --input␣
↪proteindb_from_lincRNA_canonical_sequences.fa --output decoy_proteindb.fa
```

Contributions

- Husen M. Umer ([husensofteng](https://github.com/husensofteng))
- Yafeng Zhu ([yafeng](http://github.com/yafeng))
- Enrique Audain ([enriquea](https://github.com/enriquea))
- Yasset Perez-Riverol ([ypriverol](https://github.com/ypriverol))

1.2.3 PepBedR: R package to analyze peptidoforms features in Bed Files

The PepBedR package allows users of *BED Format* file format to analyze and visualized their data and results. It facilitate two major things:

- Descriptive statistics of PepBed files: Number of unique peptides per chromosome, transcript, gene.
- Circle plots of peptide features by different Peptide properties (Modification, Uniqueness, Sample properties).

Note: The PepBedR provides set `Rscript` utilities to describe PepBed files.

Using R package

Parsing Bed file

```
bed_path <- '/home/biolinux/temp/human/pride_cluster_peptides_9606_Human_pogo.bed'

granges_peptide <- importBEDasGRange(inputFile = bed_path)

n_features <- length(granges_peptide)
message(c('Imported ', n_features, ' peptides...'))
```

Computing some basic stats from the data

The PepBedR provides a set of functions and rutines to compute descriptive statistics for each input file.

```
counts <- countsByChromosome(gr = granges_peptide, colName = 'Peptides')

merged_counts <- orderByChromosome(df = count, colName = 'Chromosome')

print(merged_counts)
```

This code will print the number of peptides per chromosome.

Getting stats for unique peptides

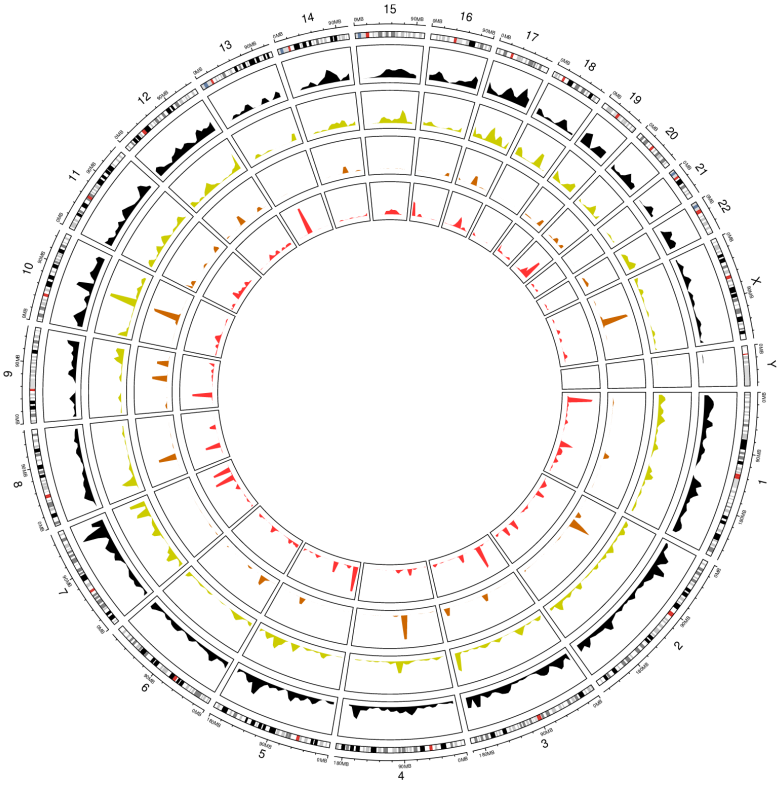
```
# Removing duplicated entries from original granges_peptide.

unique_pep <- getUniqueFeatures(granges_peptide, colFeatures = 'name')

getting unique number of features(peptides) by chromosome
counts_unique <- countsByChromosome(gr = unique_pep, colName = 'Peptides')

# ordering by chromosome
merged_counts_unique <- orderByChromosome(df = counts_unique, colName = 'Chromosome')
print(merged_counts_unique)
```

Visualizing the data

The distribution of peptides by chromosome. (blue_track: modified peptide; red_track: non-modified)

```
library(circlize)
circos.initializeWithIdeogram(species = 'hg19')
bed <- bed_df
bed_mod <- bed_mod_df
circos.genomicDensity(bed, col = c("#FF000080"), track.height = 0.1, baseline = 0)
circos.genomicDensity(bed_mod, col = c("#0000FF80"), track.height = 0.1, baseline = 0)
circos.clear()
```

Output:



Note: The distribution of peptides by chromosome. The PepBedR package use the same color code that *BED Format* to each track.

Reports

The current package provides a way to generate **pdf** reports by running the following RScript:

```
Rscript --verbose --vanilla scripts/build_pepbed_report.R -i PepGenome-Peptide-Atlas.
↪bed.gz
    -ref 'hg38' -o report_peptide_atlas.pdf
```

The build_pepbed_report.R compute compute a full report for a bed file. The paramters `-ref` is the Genome Assembly version used to perform the mapping to genome coordinates; and the `-i` and `-o` are the input bed and output pdf folder.

1.3 PGATK NF-Core Workflows

The ProteoGenomics Analysis Toolkit provides a set of workflows to perform large scale proteogenomics data analysis. All workflows are developed using nextflow and BioContainers following nf-core.

All PGATK workflows are deposited in nf-core.

1.3.1 Requirements

Starting with Nextflow

Nextflow can be used on any POSIX compatible system (Linux, OS X, etc). It requires **Bash 3.2** (or later) and **Java 8** (or later, up to 11) to be installed.

Installation, it only needs two easy steps:

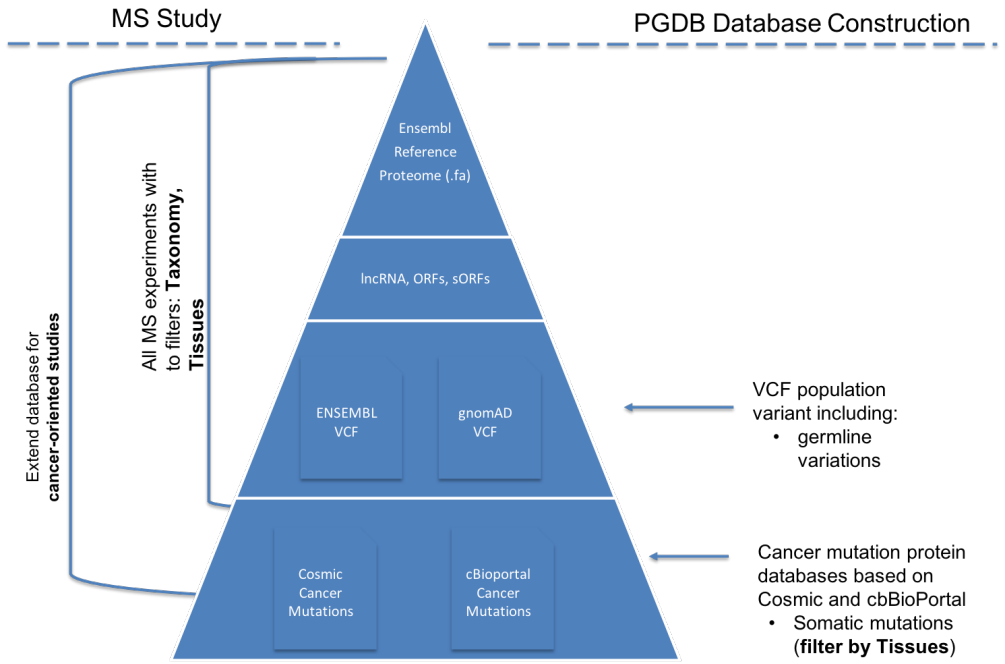
Download the executable package by copying and pasting the following command in your terminal window:

```
wget -qO- https://get.nextflow.io | bash
```

You can read more about how to setup your nf-core or nextflow enviroments.

1.3.2 PGDB: proteogenomics database generation

The ProteoGenomics DataBase (pgdb) generation pipeline is a nf-core workflow that enables the generation of custom proteogenomics databases for MS proteomics studies using the pypgatk library.



The pgdb pipelines enable to generate a variety of ENSEMBL-based proteognomics databases depending of the type of study.

Workflow usage

All workflow options can be seen by using the `--help` command:

```
N E X T F L O W  ~  version 19.04.1
Launching `main.nf` [sleepy_stonebraker] - revision: 9cff592eaf
Usage:

 The typical command for running the pipeline is as follows:

 nextflow run main.nf --taxonomy 9606 --ensembl false --gnomad false --cosmic false --
↪cbioportal false

 This command will generate a protein datbase for non-coding RNAs, pseudogenes,
 altORFs. Note the other flags are set to false.
 A final fasta file is created by merging them all and the canonical
 proteins are appended. The resulting database is stored in result/final_proteinDB.fa
 and its decoy is stored under result/decoy_final_proteinDB.fa

 Options:

 Process flags
   --ncrna [true | false]             Generate protein database from non-coding RNAs
   --pseudogenes [true | false]       Generate protein database from pseudogenes
   --altorfs [true | false]           Generate alternative ORFs from canonical␣
↪proteins
   --cbioportal [true | false]        Download cBioPortal studies and genrate protein␣
↪database
   --cosmic [true | false]            Download COSMIC files and generate protein␣
↪database
   --ensembl [true | false]           Download ENSEMBL variants and generate protein␣
↪database
   --gnomad [true | false]            Download gnomAD files and generate protein␣
↪database
   --decoy [true | false]             Append the decoy proteins to the database


 Configuration files                  By default all config files are located in the␣
↪configs
                                        directory.
   --ensembl_downloader_config        Path to configuration file for ENSEMBL download␣
↪parameters
   --ensembl_config                   Path to configuration file for parameters in␣
↪generating
                                        protein databases from ENSMEBL sequences
   --cosmic_config                    Path to configuration file for parameters in␣
↪generating
                                        protein databases from COSMIC mutations
   --cbioportal_config                Path to configuration file for parameters in␣
↪generating
                                        protein databases from cBioPortal mutations
   --protein_decoy_config             Path to configuration file for parameters used␣
↪in generating
                                        decoy databases

 Database parameters:
   --taxonomy                         Taxonomy (Taxon ID) for the species to download␣
↪ENSEMBL data,
```

(continues on next page)

(continued from previous page)

```
                                        default is 9606 for humans.
                                    For the list of supported taxonomies see:
                                        https://www.ensembl.org/info/about/species.
↪html

   --cosmic_tissue_type             Specify a tissue type to limit the COSMIC␣
↪mutations to
                                        a particular caner type (by default all tumor␣
↪types are used)
   --cbioportal_tissue_type         Specify a tissue type to limit the cBioPortal␣
↪mutations to
                                        a particular caner type (by default all tumor␣
↪types are used)
   --af_field                       Allele frequency identifier string in VCF Info␣
↪column,
                                        if no AF info is given set it to empty.
                                        For human VCF files from ENSEMBL the default␣
↪is set to MAF

 Output parameters:
   --final_database_protein         Output file name for the final database protein␣
↪fasta file
                                        under the result/ directory.
   --decoy_prefix                   String to be used as prefix for the generated␣
↪decoy sequences

   --result_file                    Output file-path for the final database, not␣
↪under the result folder.

 Data download parameters:
   --cosmic_user_name               User name (or email) for COSMIC account
   --cosmic_password                Password for COSMIC account
                                    In order to be able to download COSMIC data,␣
↪the user should
                                    provide a user and password. Please first␣
↪register in COSMIC
                                    database (https://cancer.sanger.ac.uk/cosmic/
↪register).

   --gencode_url                    URL for downloading GENCODE datafiles:
                                        gencode.v19.pc_transcripts.fa.gz and
                                        gencode.v19.annotation.gtf.gz
   --gnomad_file_url                URL for downloading gnomAD VCF file(s)

   --help                           Print this help document


␣
↪=============================================================================================
 Pipeline Tasks:
␣
↪=============================================================================================

 Get fasta proteins, cdnas, ncRNAs and gtf files from ENSEMBL (default species = 9606)
     (processes: ensembl_fasta_download, gunzip_ensembl_files, merge_cdnas)

 Generate ncRNA, psudogenes, altORFs databases
```

(continues on next page)

(continued from previous page)

```
    (processes: add_ncrna, add_pseudogenes , add_altorfs)

Generate ENSEMBL variant protein database (VCFs, default species = 9606)
    (processes: ensembl_vcf_download, gunzip_vcf_ensembl_files, check_ensembl_vcf,␣
↪ensembl_vcf_proteinDB)

Generate gnomAD variant protein database
    (processes: gencode_download, , extract_gnomad_vcf, gnomad_proteindb)

Generate COSMIC mutated protein database (default all cancer types)
    (processes: cosmic_download , gunzip_cosmic_files, cosmic_proteindb)

Generate cBioPortal mutated protein database (default all studies and all cancer␣
↪types)
    (processes: cds_GRCh37_download, download_all_cbioportal, cbioportal_proteindb)

Concatenate all generated databases
    (processes: merge_proteindbs)

Generate a decoy database from the concatenated database
    (processes: decoy)
-------------------------------------------------------------------------------
↪---
```

Seudo-genes, long non-coding RNAs

If the study attempt to identified novel **pseudo-genes**, **long non-coding RNA peptides** and proteins in Human, the users can generate the database by concatenating the ENSEMBL Human reference proteome and the novel coding regions using the following command:

```
nextflow run main.nf --taxonomy 9606 --ensembl false --gnomad false --cosmic false --
↪cbioportal false --altorfs false -profile local,standard -c nextflow.config
```

This command will attached to the reference ENSEMBL Human proteome, the **pseudo-genes** and (pipeline option –pseudogenes) and the long non-coding RNA peptides (–ncrna).

Note: Most of the options in the pipeline are enable by default. For example **–add_reference**, includes in the results database the reference proteome for the species under study.

COSMIC and cBioPortal Variants

If COSMIC variants wants to be added to the database, the following command can be used:

```
nextflow run main.nf --taxonomy 9606 --ensembl false --gnomad false --cosmic true --
↪cbioportal false --altorfs false  --cosmic_user_name username --cosmic_password␣
↪password -profile local,standard -c nextflow.config
```

Note: For COSMIC database a user and password should be provided to the pipeline to be able to download the database variants and the celllines information.

1.4 PGATK File Formats

The ProteoGenomics Analysis ToolKit is based on standard proteomics formats developed by HUPO-PSI and Genomics Standard file formats. This section is to highlight in 10 minutes the most important features of those file formats, How they are used in PGATK and you can contribute to their development.

Note: It is important to notice that this Help page only provides the fundamentals of each file format used in PGATK, for more details we provide links to the original documentation of the File format.

1.4.1 BED Format

BED

BED ***(Browser Extensible Data)** format provides a flexible way to define the data lines that are displayed in an annotation track UCSC Bed Definition. BED lines have three required fields and nine additional optional fields. The number of fields per line must be consistent throughout any single set of data in an annotation track. The order of the optional fields is binding: lower-numbered fields must always be populated if higher-numbered fields are used.

Note: If your data set is BED-like, but it is very large (over 50MB) and you would like to keep it on your own server, you should use the bigBed data format.

The **pedBed** Fields and Properties supported by PGATK:

Field (bold are required)	Description	Example
chrom	The name of the chromosome	chr3
chromStart	The starting position of the feature in the chromosome or scaffold	1000
chromEnd	The ending position of the feature in the chromosome or scaffold	5000
name	Defines the label of the BED line. GPATK annotate peptide sequence	K(Phospho)SR
score	A score between 0 and 1000.	1000
strand	Defines the strand. Either "." (=no strand) or "+" or "-".	•
thickStart	The starting position at which the feature is drawn thickly.	
thickEnd	The ending position at which the feature is drawn thickly	5000
itemRgb	An RGB value that will determine the display color of BED line color	(0,0,255)
blockCount	The number of blocks (exons) in the BED line.	
blockSizes	A comma-separated list of the block sizes.	
blockStarts	A comma-separated list of block starts.	
proteinAccession	Protein accession number	
transcriptAccession	Transcript Accession	
peptideSequence	Peptide Sequence with no PTMs added	
proteinUniqueness	Peptide uniqueness (See color code color)	
transcriptUniqueness	Peptide uniqueness (See color code color)	
genomeReferenceVersion	Genome reference version number	
psmScore	Best PSM score	
fdr	False-discovery rate	
modifications	Coma separated list of Post-translational modifications	
peptideRepeatCount	Peptide Counting	
datasetAccession	Dataset Identifier	
uri	Uniform Resource Identifier	

Hint: If the field content is to be empty the space should be field with a "."

Note: BED input files (and input received from stdin) are tab-delimited. The following types of BED files are supported by PGATK:

- **BED4**: A BED file where each feature is described by chrom, start, end, and name. (e.g. chr1 11873 14409 VLADIMIR)
- **BED6**: A BED file where each feature is described by chrom, start, end, name, score, and strand. (e.g. chr1 11873 14409 VLADIMIR 0 +)

- **BED12**: A BED file where each feature is described by all twelve columns listed above. (Default option in all tools)
- **BED12+11**: A complete Bed file including **required** fields and optionals.

Color

Uniqueness Colors:

Colour	Description
	Peptide is unique to single gene AND single transcript
	Peptide is unique to single gene BUT shared between multiple transcripts
	Peptide is shared between multiple genes

Modified Peptides Colors:

Like BED but containing the location of the post-translational modification on the genome. Thick parts of the peptide blocks indicate the position of the post-translational modification on a single amino acid (short thick block) while longer blocks indicate the occurrence of the first and last post-translational modification and residues in between. In the PTMBED the colour code is changed to indicate the type of modification.

Colour	Post-translational Modification
	Phosphorylation (phospho)
	Acetylation (acetyl)
	Amidation (amidated)
	Oxidation (oxidation)
	Oxidation (oxidation)
	Methylation (methyl)
	Ubiquitinylation (glygly; gg)
	Sulfation (sulfo)
	Palmitoylation (palmitoyl)
	Formylation (formyl)
	Deamidation (deamidated)
	Any other post-translational modification

1.4.2 Additional Files formats

Peptide Atlas Peptide List

PeptideAtlas released every month/year a list of peptides that has been found/identified by MS/MS (see the list here). The PGATK support the output list as input of some of the tools such as pepgenome .

Column	Field	Description
1	peptide_accession	Peptide Accession (PAp06389395)
2	peptide_sequence	Peptide Sequence
3	best_probability	Best Peptide Probability
4	n_observations	Spectral counting
		More properties not used

Hint: For our pipelines and tools the order of the column is important.

Note: A full pipeline to map the PeptideAltas peptide evidences to Genome Coordinates can be found here.

1.5 About

The ProteoGeonomics Analysis Toolkit is developed by the following people:

- Yasset Perez-Riverol (EMBL-EBI)
- Enrique Audain (Kiel University)
- Chakradhar Reddy Bandla

1.5.1 Support

You can ask support questions here: https://github.com/bigbio/pgatk/issues