pVACtools Documentation

Release 1.5.7

Jasreet Hundal, Susanna Kiwala, Aaron Graubert, Jason Walker, C

Apr 22, 2020

Contents

1 pVACseq **3**
- 1.1 Features . . . 3
- 1.2 Input File Preparation . . . 5
- 1.3 Getting Started . . . 13
- 1.4 Usage . . . 14
- 1.5 Output Files . . . 17
- 1.6 Filtering Commands . . . 21
- 1.7 Additional Commands . . . 25
- 1.8 Optional Downstream Analysis Tools . . . 26
- 1.9 Common Errors . . . 28
- 1.10 Frequently Asked Questions . . . 30

2 pVACbind **33**
- 2.1 Prerequisites . . . 33
- 2.2 Getting Started . . . 33
- 2.3 Usage . . . 34
- 2.4 Output Files . . . 36
- 2.5 Filtering Commands . . . 37
- 2.6 Additional Commands . . . 39

3 pVACfuse **41**
- 3.1 Prerequisites . . . 41
- 3.2 Getting Started . . . 42
- 3.3 Usage . . . 43
- 3.4 Output Files . . . 45
- 3.5 Filtering Commands . . . 48
- 3.6 Additional Commands . . . 50
- 3.7 Optional Downstream Analysis Tools . . . 51

4 pVACvector **53**
- 4.1 Prerequisites . . . 53
- 4.2 Getting Started . . . 54
- 4.3 Usage . . . 54
- 4.4 Additional Commands . . . 56
- 4.5 Output Files . . . 58

5 pVACviz **59**

5.1 Installation . . . 59
5.2 Running pVACviz . . . 60
5.3 pVACapi Directories . . . 60
5.4 Starting Processes . . . 61
5.5 Managing Processes . . . 63
5.6 Visualizing Processes . . . 66
5.7 pVACapi Troubleshotting . . . 70

6 Installation **71**
6.1 Installing IEDB binding prediction tools (strongly recommended) . . . 72
6.2 Installing MHCflurry . . . 73
6.3 Installing MHCnuggets . . . 74
6.4 PostgreSQL . . . 74
6.5 Docker and CWL . . . 74

7 Tools Used By pVACtools **77**
7.1 IEDB (Immune Epitope Database) . . . 77
7.2 MHCflurry . . . 77
7.3 MHCnuggets . . . 78
7.4 NetChop . . . 78
7.5 NetMHCstabpan . . . 78
7.6 Vaxrank . . . 78

8 Frequently Asked Questions **79**

9 Release Notes **81**
9.1 Version 1.0 . . . 81
9.2 Version 1.1 . . . 83
9.3 Version 1.2 . . . 84
9.4 Version 1.3 . . . 85
9.5 Version 1.4 . . . 87
9.6 Version 1.5 . . . 88

10 Citations **93**

11 Contact **95**

12 Mailing List **97**

13 New in release 1.5.7 **99**

14 New in version 1.5 **101**

15 Citations **103**

16 Source code **105**

17 License **107**

pVACtools is a cancer immunotherapy tools suite consisting of the following tools:

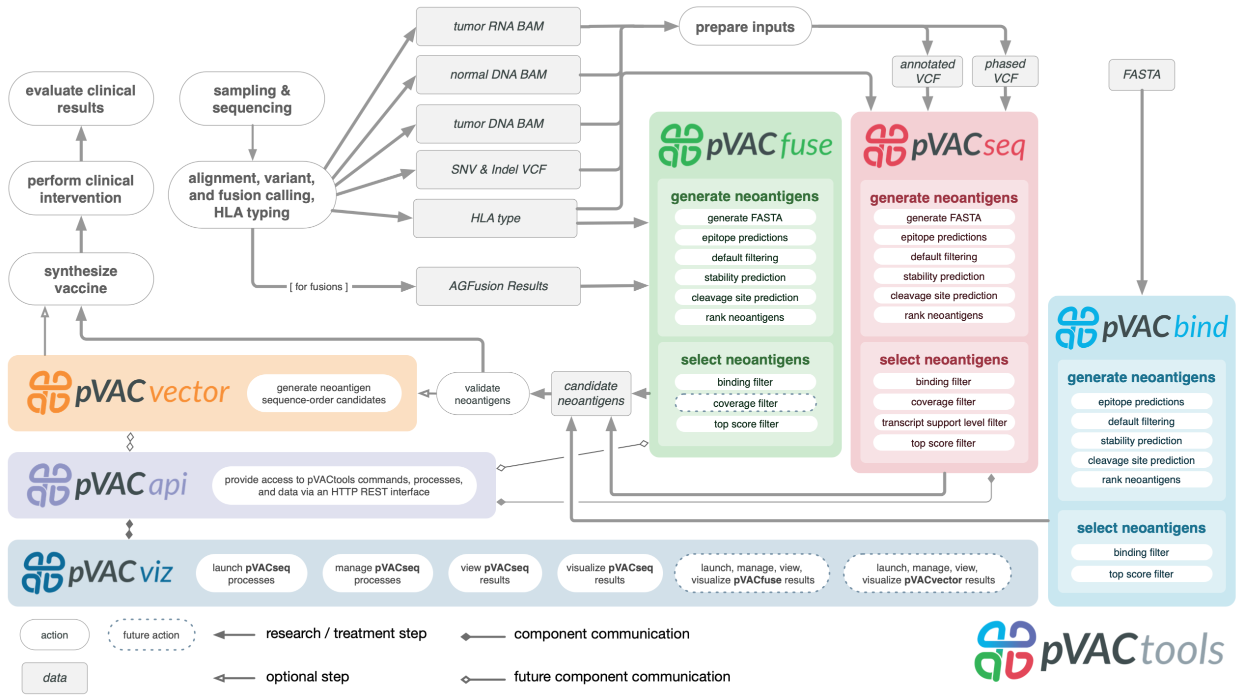
pVACseq A cancer immunotherapy pipeline for identifying and prioritizing neoantigens from a VCF file.

pVACbind A cancer immunotherapy pipeline for identifying and prioritizing neoantigens from a FASTA file.

pVACfuse A tool for detecting neoantigens resulting from gene fusions.

pVACvector A tool designed to aid specifically in the construction of DNA-based cancer vaccines.

pVACviz A browser-based user interface that assists users in launching, managing, reviewing, and visualizing the results of pVACtools processes.



CHAPTER 1

pVACseq

pVACseq is a cancer immunotherapy pipeline for the identification of **p**ersonalized **V**ariant **A**ntigens by **C**ancer **Seq**uencing (pVACseq) that integrates tumor mutation and expression data (DNA- and RNA-Seq). It enables cancer immunotherapy research by using massively parallel sequence data to predicting tumor-specific mutant peptides (neoantigens) that can elicit anti-tumor T cell immunity. It is being used in studies of checkpoint therapy response and to identify targets for personalized cancer vaccines and adoptive T cell therapies. For more general information, see the manuscript published in Genome Medicine.

1.1 Features

SNV and Indel support

pVACseq offers epitope binding predictions for missense, in-frame insertion, in-frame deletion, protein-altering, and frameshift mutations.

VCF support

pVACseq uses a VCF file as its input. This VCF file must contain sample genotype information and be annotated with the Ensembl Variant Effect Predictor (VEP). See the *Input File Preparation* section for more information.

No local install of epitope prediction software needed

pVACseq utilizes the IEDB RESTful web interface. This means that none of the underlying prediction software, like NetMHC, needs to be installed locally.

Warning: We only recommend using the RESTful API for small requests. If you use the RESTful API to process large VCFs or to make predictions for many alleles, epitope lengths, or prediction algorithms, you might overload their system. This can result in the blacklisting of your IP address by IEDB, causing 403 errors when trying to use the RESTful API. In that case please open a ticket with IEDB support to have your IP address removed from the IEDB blacklist.

Support for local installation of the IEDB Analysis Resources

pVACseq provides the option of using a local installation of the IEDB MHC class I and class II binding prediction tools.

Warning: Using a local IEDB installation is strongly recommended for larger datasets or when the making predictions for many alleles, epitope lengths, or prediction algorithms. More information on how to install IEDB locally can be found on the *Installation* page (note: the pvactools docker image now contains IEDB).

MHC Class I and Class II predictions

Both MHC Class I and Class II predictions are supported. Simply choose the desired prediction algorithms and HLA alleles during processing and Class I and Class II prediction results will be written to their own respective subdirectories in your output directory.

By using the IEDB RESTful web interface, pVACseq leverages their extensive support of different prediction algorithms.

In addition to IEDB-supported prediction algorithms, we've also added support for MHCflurry and MHCnuggets.

MHC Class I Prediction Algorithm	Version
NetMHCpan	4.0
NetMHC	4.0
NetMHCcons	1.1
PickPocket	1.1
SMM	1.0
SMMPMBEC	1.0
MHCflurry	
MHCnuggets	

MHC Class II Prediction Algorithm	Version
NetMHCIIpan	3.2
SMMalign	1.1
NNalign	2.3
MHCnuggets	

Comprehensive filtering

Automatic filtering on the binding affinity ic50 (nm) value narrows down the results to only include "good" candidate peptides. The binding filter threshold can be adjusted by the user for each pVACseq run. pVACseq also support the option of filtering on allele-specific binding thresholds as recommended by IEDB. Additional filtering on the binding affitinity can be manually done by the user by running the *standalone binding filter* on the filtered result file to narrow down the candidate epitopes even further or on the unfiltered all_epitopes file to apply different cutoffs.

Readcount and expression data are extracted from an annotated VCF to automatically filter with adjustable thresholds on depth, VAF, and/or expression values. The user can also manually run the *standalone coverage filter* to further narrow down their results from the filtered output file.

If the input VCF is annotated with Ensembl transcript support levels (TSLs), pVACseq will filter on the transcript support level to only keep high-confidence transcripts of level 1. This filter can also be run *standalone*.

As a last filtering step, pVACseq applies the top score filter to only keep the top scoring epitope for each variant. As with all previous filters, this filter can also be run *standalone*.

Ranking of candidate neoepitopes

Filtered neoepitopes are *ranked* based on the binding affinity, fold change between mutant and wildtype binding affinity (agretopicity), gene expression, RNA and DNA VAF.

Incorporation of proximal germline and somatic variants

To incorporate proximal variants into the neoepitope predictions, users can provide a *phased VCF of proximal variants* as an input to their pVACseq runs. This VCF is then used to incorporate amino acid changes of nearby variants that are in-phase with a somatic variant of interest. This results in corrected mutant and wildtype protein sequences that account for proximal variants when MHC binding predictions are performed.

NetChop and NetMHCstab integration

Cleavage position predictions are added with optional processing through NetChop.

Stability predictions can be added if desired by the user. These predictions are obtained via NetMHCstabpan.

1.2 Input File Preparation

The main input file to the pVACseq pipeline is a VCF file. The VCF needs to contain sample genotype information (`GT` field). The VCF needs to be annotated with VEP to add transcript information.

If filtering on variant allele fractions (VAFs), depth, and expression values is desired, the VCF also needs to be annotated with this data.

Refer to the following sections for instructions on how to annotate your VCF with these data and how to produce a VCF for proximal variant analysis.

1.2.1 Annotating your VCF with VEP

The input to the pVACseq pipeline is a VEP-annotated VCF. This will add consequence, transcript, and gene information to your VCF.

Installing VEP

1. To download and install the VEP command line tool follow the VEP installation instructions.
2. We recommend the use of the VEP cache for your annotation. The VEP cache can be downloaded following these VEP cache installation instructions. Please ensure that the Ensembl cache version matches the reference build and Ensembl version used in other parts of your analysis (e.g. for RNA-seq gene/transcript abundance estimation).
3. Download the VEP plugins from the GitHub repository by cloning the repository:

```
git clone https://github.com/Ensembl/VEP_plugins.git
```

4. *Copy the Wildtype plugin* provided with the pVACseq package to the folder with the other VEP plugins by running the following command:

```
pvacseq install_vep_plugin <VEP plugins directory>
```

Running VEP

Example VEP Command

```
./vep \
--input_file <input VCF> --output_file <output VCF> \
--format vcf --vcf --symbol --terms SO --tsl\
--hgvs --fasta <reference build FASTA file location> \
--offline --cache [--dir_cache <VEP cache directory>] \
--plugin Downstream --plugin Wildtype \
[--dir_plugins <VEP_plugins directory>] [--pick] [--transcript_version]
```

Required VEP Options

```
--format vcf
--vcf
--symbol
--terms SO
--tsl
--hgvs
--fasta <reference build FASTA location>
--offline
--cache
--plugin Downstream
--plugin Wildtype
```

- The `--format vcf` option specifies that the input file is in VCF format.
- The `--vcf` option will result in the output being written in VCF format.
- The `--symbol` option will include gene symbol in the annotation.
- The `--terms SO` option will result in Sequence Ontology terms being used for the consequences.
- The `--tsl` option adds transcript support level information to the annotation.
- The `--hgvs` option will result in HGVS identifiers being added to the annotation.
- Using the `--hgvs` option requires the usage of the `--fasta` argument to specify the location of the reference genome build FASTA file.
- The `--offline` option will eliminate all network connections for speed and/or privacy.
- The `--cache` option will result in the VEP cache being used for annotation.
- The `--plugin Downstream` option will run the Downstream plugin which will compute the downstream protein sequence after a frameshift.
- The `--plugin Wildtype` option will run the Wildtype plugin which will include the transcript protein sequence in the annotation.

Useful VEP Options

```
--dir_cache <VEP cache directory>
--dir_plugins <VEP_plugins directory>
--pick
--transcript_version
```

- The `--dir_cache <VEP cache directory>` option may be needed if the VEP cache was downloaded to a different location than the default. The default location of the VEP cache is at `$HOME/.vep`.
- The `--dir_plugins <VEP_plugins directory>` option may need to be set depending on where the VEP_plugins were installed to.
- The `--pick` option might be useful to limit the annotation to the "top" transcript for each variant (the one for which the most dramatic consequence is predicted). Otherwise, VEP will annotate each variant with all possible transcripts. pVACseq will provide predictions for all transcripts in the VEP CSQ field. Running VEP without the `--pick` option can therefore drastically increase the runtime of pVACseq.
- The `--transcript_version` option will add the transcript version to the transcript identifiers. This option might be needed if you intend to annotate your VCF with expression information. Particularly if your expression estimation tool uses versioned transcript identifiers (e.g. ENST00000256474.2).

Additional VEP options that might be desired can be found here.

1.2.2 Adding coverage data to your VCF

pVACseq is able to parse coverage information directly from the VCF. The expected annotation format is outlined below.

Type	VCF Sample	Format Fields
Tumor DNA Coverage	single-sample VCF or `sample_name`	`AD`, `DP`, and `AF`
Tumor RNA Coverage	single-sample VCF or `sample_name`	`RAD`, `RDP`, and `RAF`
Normal DNA Coverage	`--normal-sample-name`	`AD`, `DP`, and `AF`

Tumor DNA Coverage

If the VCF is a single-sample VCF, pVACseq assumes that this sample is the tumor sample. If the VCF is a multi-sample VCF, pVACseq will look for the sample using the `sample_name` parameter and treat that sample as the tumor sample.

For this tumor sample, the tumor DNA depth is determined from the `DP` format field. The tumor DNA VAF is determined from the `AF` field. If the VCF does not contain a `AF` format field, the tumor DNA VAF is calculated from the `AD` and `DP` fields by dividing the allele count by the total read depth.

Tumor RNA Coverage

If the VCF is a single-sample VCF, pVACseq assumes that this sample is the tumor sample. If the VCF is a multi-sample VCF, pVACseq will look for the sample using the `sample_name` parameter and treat that sample as the tumor sample.

For this tumor sample, the tumor RNA depth is determined from the `RDP` format field. The tumor RNA VAF is determined from the `RAF` field. If the VCF does not contain a `RAF` format field, the tumor RNA VAF is calculated from the `RAD` and `RDP` fields by dividing the allele count by the total read depth.

Normal DNA Coverage

To parse normal DNA coverage information, the input VCF to pVACseq will need to be a multi-sample (tumor/normal) VCF, with one sample being the tumor sample, and the other the matched normal sample. The tumor sample is identified by the `sample_name` parameter while the normal sample can be specified with `--normal-sample-name` option.

For this normal sample, the normal DNA depth is determined from the `DP` format field. The normal DNA VAF is determined from the `AF` field. If the VCF does not contain a `AF` format field, the normal DNA VAF is calculated from the `AD` and `DP` fields by dividing the allele count by the total read depth.

Using the vcf-readcount-annotator to add coverage information to your VCF

Some variant callers will already have added coverage information to your VCF. However, if your VCF doesn't contain coverage information or if you need to add coverage information for additional samples or for RNA-seq data, you can use the `vcf-readcount-annotator` to do so. The `vcf-readcount-annotator` will take the output from bam-readcount and use it to add readcounts to your VCF.

bam-readcount needs to be run separately for snvs and indels so it is recommended to first split multi-allelic sites by using a tool such as `vt decompose`.

Installing vt

The `vt` tool suite can be installed by following the instructions on their page.

Installing bam-readcount

The `vcf-readcount-annotator` will add readcount information from bam-readcount output files to your VCF. Therefore, you will first need to run bam-readcount to obtain a file of readcounts for your variants.

Follow the installation instructions on the bam-readcount GitHub page.

Installing the vcf-readcount-annotator

The `vcf-readcount-annotator` is part of the `vcf-annotation-tools` package. Please visit vatools.org for more details on this package. You can install this package by running:

```
pip install vcf-annotation-tools
```

Running vt decompose

Example vt decompose command

```
vt decompose -s <input_vcf> -o <decomposed_vcf>
```

Running bam-readcount

bam-readcount uses a bam file and site list regions file as input. The site lists are created from your decomposed VCF, one for snvs and one for indels. Snvs and indels are then run separately through bam-readcount using the same bam. Indel regions must be run in a special insertion-centric mode.

Example bam-readcount command

```
bam-readcount -f <reference_fasta> -l <site_list> <bam_file> [-i] [-b 20]
```

The `-i` option must be used when running the indels site list in order to process indels in insertion-centric mode.

A minimum base quality of 20 is recommended which can be enabled using the `-b 20` option.

The `mgibio/bam_readcount_helper-cwl` Docker container contains a `bam_readcount_helper.py` script that will create the snv and indel site list files from a VCF and run bam-readcount. Information on that Docker container can be found here: dockerhub mgibio/bam_readcount_helper-cwl.

Example bam_readcount_helper.py command

```
/usr/bin/python /usr/bin/bam_readcount_helper.py \
<decomposed_vcf> <sample_name> <reference_fasta> <bam_file> <output_dir>
```

This will write two bam-readcount files to the `<output_dir>`: `<sample_name>_bam_readcount_snv.tsv` and `<sample_name>_bam_readcount_indel.tsv`, containing readcounts for the snv and indel positions, respectively.

Running the vcf-readcount-annotator

The readcounts for snvs and indels are then added to your VCF separately, by running the `vcf-readcount-annotator` twice.

Example vcf-readcount-annotator commands

```
vcf-readcount-annotator <decomposed_vcf> <snv_bam_readcount_file> <DNA|RNA> \
-s <sample_name> -t snv -o <snv_annotated_vcf>

vcf-readcount-annotator <snv_annotated_vcf> <indel_bam_readcount_file> <DNA|RNA> \
-s <sample_name> -t indel -o <annotated_vcf>
```

The data type `DNA` or `RNA` identifies whether you are annotating DNA or RNA readcount. DNA readcount annotations will be written to the `AD/DP/AF` format fields while RNA readcount annotations will be written to the `RAD/RDP/RAF` format fields. Please see the VAtools documentation for more information.

1.2.3 Adding expression data to your VCF

pVACseq is able to parse coverage and expression information directly from the VCF. The expected annotation format is outlined below.

Type	VCF Sample	Format Fields
Transcript Expression	single-sample VCF or `sample_name`	`TX`
Gene Expression	single-sample VCF or `sample_name`	`GX`

Transcript Expression

If the VCF is a single-sample VCF, pVACseq assumes that this sample is the tumor sample. If the VCF is a multi-sample VCF, pVACseq will look for the sample using the `sample_name` parameter and treat that sample as the tumor sample.

For this tumor sample the transcript expression is determined from the `TX` format field. The `TX` format field is a comma-separated list of per-transcript expression values, where each individual transcript expression is listed as `expression_id|expression_value`, e.g. `ENST00000215794|2.35912,ENST00000215795|0.2`. The `expression_id` needs to match the `Feature` field of the VEP `CSQ` annotation. In other words, your expression abundance estimation should have been performed with the same transcript annotation version that you used to annotate your variants with VEP (e.g. Ensembl v95).

Gene Expression

If the VCF is a single-sample VCF, pVACseq assumes that this sample is the tumor sample. If the VCF is a multi-sample VCF, pVACseq will look for the sample using the `sample_name` parameter and treat that sample as the tumor sample.

For this tumor sample the gene expression is determined from the `GX` format field. The `GX` format field is a comma-separated list of per-gene expression values, where each individual gene expression is listed as `gene_id|expression_value`, e.g. `ENSG00000184979|2.35912`. The `gene_id` needs to match the `Gene` field of the VEP `CSQ` annotation.

Using the vcf-expression-annotator to add expression information to your VCF

The `vcf-expression-annotator` will add expression information to your VCF. It will accept expression data from various tools. Currently it supports Cufflinks, Kallisto, StringTie, as well as a custom option for any tab-delimited file.

Installing the vcf-expression-annotator

The `vcf-expression-annotator` is part of the `vcf-annotation-tools` package (vatools.org). You can install this package by running:

```
pip install vcf-annotation-tools
```

Running the vcf-expression-annotator

You can now use the output file from your expression caller to add expression information to your VCF:

```
vcf-expression-annotator input_vcf expression_file␣
↪kallisto|stringtie|cufflinks|custom gene|transcript
```

The data type `gene` or `transcript` identifies whether you are annotating transcript or gene expression data. Transcript expression annotations will be written to the `TX` format field while gene expression annotations will be written to the `GX` format field. Please see the VAtools documentation for more information.

1.2.4 Creating a phased VCF of proximal variants

By default, pVACseq will evaluate all somatic variants in the input VCF in isolation. As a result, if a somatic variant of interest has other somatic or germline variants in proximity, the calculated wildtype and mutant protein sequences might be incorrect because the amino acid changes of those proximal variants were not taken into account.

To solve this problem, we added a new option to pVACseq in the `pvactools` release 1.1. This option, `--phased-proximal-variants-vcf`, can be used to provide the path to a phased VCF of proximal variants in addition to the normal input VCF. This VCF is then used to incorporate amino acid changes of nearby variants that are in-phase to a somatic variant of interest. This results in corrected mutant and wildtype protein sequences that account for proximal variants.

At this time, this option only handles missense proximal variants but we are working on a more comprehensive approach to this problem.

Note that if you do not perform the proximal variants step, you should manually review the sequence data for all candidates (e.g. in IGV) for proximal variants and either account for these manually, or eliminate these candidates. Failure to do so may lead to inclusion of incorrect peptide sequences.

How to create the phased VCF of proximal variants

Input files

- `tumor.bam`: A BAM file of tumor reads
- `somatic.vcf`: A VCF of somatic variants
- `germline.vcf`: A VCF of germline variants
- `reference.fa`: The reference FASTA file

Required tools

- Picard
- GATK
- bgzip
- tabix

Create the reference dictionary

```
java -jar picard.jar CreateSequenceDictionary \
R=reference.fa \
O=reference.dict
```

Update sample names

The sample names in the `tumor.bam`, the `somatic.vcf`, and the `germline.vcf` need to match. If they don't you need to edit the sample names in the VCF files to match the tumor BAM file.

Combine somatic and germline variants using GATK's CombineVariants

```
/usr/bin/java -Xmx16g -jar /opt/GenomeAnalysisTK.jar \
-T CombineVariants \
-R reference.fa \
--variant germline.vcf \
--variant somatic.vcf \
-o combined_somatic_plus_germline.vcf \
--assumeIdenticalSamples
```

Sort combined VCF using Picard

```
/usr/bin/java -Xmx16g -jar /opt/picard/picard.jar SortVcf \
I=combined_somatic_plus_germline.vcf \
O=combined_somatic_plus_germline.sorted.vcf \
SEQUENCE_DICTIONARY=reference.dict
```

Phase variants using GATK's ReadBackedPhasing

```
/usr/bin/java -Xmx16g -jar /opt/GenomeAnalysisTK.jar \
-T ReadBackedPhasing \
-R reference.fa \
-I tumor.bam \
--variant combined_somatic_plus_germline.sorted.vcf \
-L combined_somatic_plus_germline.sorted.vcf \
-o phased.vcf
```

bgzip and index the phased VCF

```
bgzip -c phased.vcf > phased.vcf.gz
tabix -p vcf phased.vcf.gz
```

The resulting `phased.vcf.gz` file can be used as the input to the `--phased-proximal-variants-vcf` option.

bgzip and index the input VCF

In order to use the `--phased-proximal-variants-vcf` option you will also need to bgzip and index your main input VCF.

```
bgzip -c input.vcf > input.vcf.gz
tabix -p vcf input.vcf.gz
```

1.3 Getting Started

pVACseq provides a set of example data to show the expected input and output files. You can download the data set by running the `pvacseq download_example_data` *command*.

The example data output can be reproduced by running the following command:

```
pvacseq run \
<example_data_dir>/input.vcf \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
<output_dir> \
-e 8,9,10
```

A detailed description of all command options can be found on the *Usage* page.

1.3.1 Running pVACseq using Docker

A pVACtools Docker image is available on DockerHub using the `griffitlab/pvactools` tag. After installing Docker you can start an interactive Docker instance by running the following command:

```
docker run -it griffithlab/pvactools
```

Version-specific images are available and can be run like so:

```
docker run -it griffithlab/pvactools:<version>
```

In order to have access to your local data inside of the Docker container you will need to mount a local volume inside of the container. This is done using the `-v` flag. For example, you can mount your `/local/path/to/example_data_dir` in your container like so:

```
docker run -v /local/path/to/example_data_dir:/pvactools_example_data -it griffithlab/
↪pvactools
```

This will mount the `example_data_dir` inside the container as the `/pvacseq_example_data` directory. When you are inside of the container you will now have access to all of the data that was inside of the `example_data_dir` from the `/pvaseq_example_data` directory.

You will need to do the same thing for your `/local/path/to/output_dir` so that any output written by pVACseq will be accessible from your machine outside of your Docker container.

```
docker run -v /local/path/to/example_data_dir:/pvacseq_example_data -v /local/path/to/
↪output_dir:/pvacseq_output_data -it griffithlab/pvactools
```

You can now run your pVACseq command like so:

```
pvacseq run \
/pvacseq_example_data/input.vcf \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
/pvacseq_output_data \
-e 8,9,10
--iedb-install-directory /opt/iedb
```

The output from your pVACseq run can be found under `/pvacseq_output_data` inside of the container and `/local/path/to/output_dir` on your local machine.

Please note that our Docker container already includes installations of the IEDB class I and class II tools at `/opt/iedb` (`--iedb-install-directory /opt/iedb`).

1.4 Usage

Warning: Using a local IEDB installation is strongly recommended for larger datasets or when the making predictions for many alleles, epitope lengths, or prediction algorithms. More information on how to install IEDB locally can be found on the *Installation* page.

```
usage: pvacseq run [-h] [-e EPITOPE_LENGTH]
                   [--iedb-install-directory IEDB_INSTALL_DIRECTORY]
                   [-b BINDING_THRESHOLD]
                   [--allele-specific-binding-thresholds] [-m {lowest,median}]
                   [-r IEDB_RETRIES] [-k] [-t N_THREADS]
                   [--net-chop-method {cterm,20s}] [--netmhc-stab]
                   [--net-chop-threshold NET_CHOP_THRESHOLD]
                   [-a {sample_name}] [-s FASTA_SIZE] [--exclude-NAs]
                   [-l PEPTIDE_SEQUENCE_LENGTH]
                   [-d DOWNSTREAM_SEQUENCE_LENGTH]
                   [--normal-sample-name NORMAL_SAMPLE_NAME]
                   [-p PHASED_PROXIMAL_VARIANTS_VCF] [-c MINIMUM_FOLD_CHANGE]
                   [--normal-cov NORMAL_COV] [--tdna-cov TDNA_COV]
                   [--trna-cov TRNA_COV] [--normal-vaf NORMAL_VAF]
                   [--tdna-vaf TDNA_VAF] [--trna-vaf TRNA_VAF]
                   [--expn-val EXPN_VAL]
                   [--maximum-transcript-support-level {1,2,3,4,5}]
                   [--pass-only]
                   input_file sample_name allele
                   {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}
                   [{MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign} ...]
                   output_dir

positional arguments:
  input_file            A VEP-annotated single- or multi-sample VCF containing
```

(continues on next page)

(continued from previous page)

```
                        genotype, transcript, Wildtype protein sequence, and
                        Downstream protein sequence information.The VCF may be
                        gzipped (requires tabix index).
  sample_name           The name of the tumor sample being processed. When
                        processing a multi-sample VCF the sample name must be
                        a sample ID in the input VCF #CHROM header line.
  allele                Name of the allele to use for epitope prediction.
                        Multiple alleles can be specified using a comma-
                        separated list. For a list of available alleles, use:
                        `pvacseq valid_alleles`.
  {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,
↪PickPocket,SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use. Multiple
                        prediction algorithms can be specified, separated by
                        spaces.
  output_dir            The directory for writing all result files.

optional arguments:
  -h, --help            show this help message and exit
  -e EPITOPE_LENGTH, --epitope-length EPITOPE_LENGTH
                        Length of subpeptides (neoepitopes) to predict.
                        Multiple epitope lengths can be specified using a
                        comma-separated list. Typical epitope lengths vary
                        between 8-11. Required for Class I prediction
                        algorithms. (default: None)
  --iedb-install-directory IEDB_INSTALL_DIRECTORY
                        Directory that contains the local installation of IEDB
                        MHC I and/or MHC II. (default: None)
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
                        allele-specific binding thresholds run `pvacseq
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the best MT Score and Corresponding Fold Change
                        (i.e. the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the median MT Score and Median
                        Fold Change (i.e. the median MT ic50 binding score and
                        fold change of all chosen prediction methods).
                        (default: median)
  -r IEDB_RETRIES, --iedb-retries IEDB_RETRIES
                        Number of retries when making requests to the IEDB
                        RESTful web interface. Must be less than or equal to
                        100. (default: 5)
  -k, --keep-tmp-files  Keep intermediate output files. This might be useful
                        for debugging purposes. (default: False)
  -t N_THREADS, --n-threads N_THREADS
                        Number of threads to use for parallelizing peptide-MHC
                        binding prediction calls. (default: 1)
  --net-chop-method {cterm,20s}
```

(continues on next page)

(continued from previous page)

```
                        NetChop prediction method to use ("cterm" for C term
                        3.0, "20s" for 20S 3.0). C-term 3.0 is trained with
                        publicly available MHC class I ligands and the authors
                        believe that is performs best in predicting the
                        boundaries of CTL epitopes. 20S is trained with in
                        vitro degradation data. (default: None)
  --netmhc-stab         Run NetMHCStabPan after all filtering and add
                        stability predictions to predicted epitopes. (default:
                        False)
  --net-chop-threshold NET_CHOP_THRESHOLD
                        NetChop prediction threshold (increasing the threshold
                        results in better specificity, but worse sensitivity).
                        (default: 0.5)
  -a {sample_name}, --additional-report-columns {sample_name}
                        Additional columns to output in the final report. If
                        sample_name is chosen, this will add a column with the
                        sample name in every row of the output. This can be
                        useful if you later want to concatenate results from
                        multiple individuals into a single file. (default:
                        None)
  -s FASTA_SIZE, --fasta-size FASTA_SIZE
                        Number of FASTA entries per IEDB request. For some
                        resource-intensive prediction algorithms like
                        Pickpocket and NetMHCpan it might be helpful to reduce
                        this number. Needs to be an even number. (default:
                        200)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
                        False)
  -l PEPTIDE_SEQUENCE_LENGTH, --peptide-sequence-length PEPTIDE_SEQUENCE_LENGTH
                        Length of the peptide sequence to use when creating
                        the FASTA. (default: 21)
  -d DOWNSTREAM_SEQUENCE_LENGTH, --downstream-sequence-length DOWNSTREAM_SEQUENCE_
↪LENGTH
                        Cap to limit the downstream sequence length for
                        frameshifts when creating the FASTA file. Use 'full'
                        to include the full downstream sequence. (default:
                        1000)
  --normal-sample-name NORMAL_SAMPLE_NAME
                        In a multi-sample VCF, the name of the matched normal
                        sample. (default: None)
  -p PHASED_PROXIMAL_VARIANTS_VCF, --phased-proximal-variants-vcf PHASED_PROXIMAL_
↪VARIANTS_VCF
                        A VCF with phased proximal variant information. Must
                        be gzipped and tabix indexed. (default: None)
  -c MINIMUM_FOLD_CHANGE, --minimum-fold-change MINIMUM_FOLD_CHANGE
                        Minimum fold change between mutant (MT) binding score
                        and wild-type (WT) score (fold change = WT/MT). The
                        default is 0, which filters no results, but 1 is often
                        a sensible choice (requiring that binding is better to
                        the MT than WT peptide). This fold change is sometimes
                        referred to as a differential agretopicity index.
                        (default: 0.0)
  --normal-cov NORMAL_COV
                        Normal Coverage Cutoff. Only sites above this read
                        depth cutoff will be considered. (default: 5)
  --tdna-cov TDNA_COV   Tumor DNA Coverage Cutoff. Only sites above this read
                        depth cutoff will be considered. (default: 10)
```

(continues on next page)

(continued from previous page)

```
  --trna-cov TRNA_COV   Tumor RNA Coverage Cutoff. Only sites above this read
                        depth cutoff will be considered. (default: 10)
  --normal-vaf NORMAL_VAF
                        Normal VAF Cutoff. Only sites BELOW this cutoff in
                        normal will be considered. (default: 0.02)
  --tdna-vaf TDNA_VAF   Tumor DNA VAF Cutoff. Only sites above this cutoff
                        will be considered. (default: 0.25)
  --trna-vaf TRNA_VAF   Tumor RNA VAF Cutoff. Only sites above this cutoff
                        will be considered. (default: 0.25)
  --expn-val EXPN_VAL   Gene and Transcript Expression cutoff. Only sites
                        above this cutoff will be considered. (default: 1.0)
  --maximum-transcript-support-level {1,2,3,4,5}
                        The threshold to use for filtering epitopes on the
                        Ensembl transcript support level (TSL). Keep all
                        epitopes with a transcript support level <= to this
                        cutoff. (default: 1)
  --pass-only           Only process VCF entries with a PASS status. (default:
                        False)
```

1.5 Output Files

The pVACseq pipeline will write its results in separate folders depending on which prediction algorithms were chosen:

- `MHC_Class_I`: for MHC class I prediction algorithms
- `MHC_Class_II`: for MHC class II prediction algorithms
- `combined`: If both MHC class I and MHC class II prediction algorithms were run, this folder combines the neoeptiope predictions from both

Each folder will contain the same list of output files (listed in the order created):

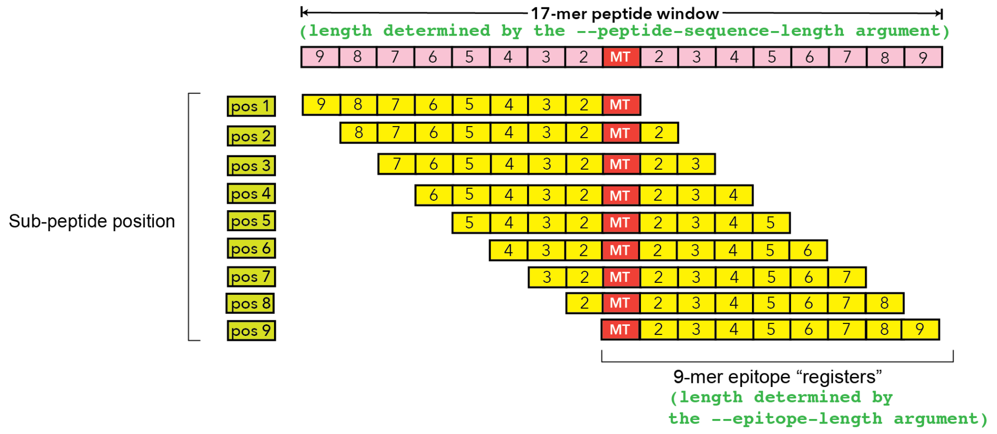
File Name	Description
`<sample_name>.tsv`	An intermediate file with variant, transcript, coverage, vaf, and expression information parsed from the input files.
`<sample_name>.tsv_<chunks>` (multiple)	The above file but split into smaller chunks for easier processing with IEDB.
`<sample_name>.all_epitopes.tsv`	A list of all predicted epitopes and their binding affinity scores, with additional variant information from the `<sample_name>.tsv`.
`<sample_name>.filtered.tsv`	The above file after applying all filters, with cleavage site and stability predictions added.
`<sample_name>.filtered.condensed.ranked.tsv`	A condensed version of the filtered TSV with only the most important columns remaining, with a priority score for each neoepitope candidate added.

1.5.1 all_epitopes.tsv and filtered.tsv Report Columns

Column Name	Description
`Chromosome`	The chromosome of this variant
`Start`	The start position of this variant in the zero-base
`Stop`	The stop position of this variant in the zero-base
`Reference`	The reference allele
`Variant`	The alt allele
`Transcript`	The Ensembl ID of the affected transcript
`Transcript Support Level`	The transcript support level (TSL) of the affecte
`Ensembl Gene ID`	The Ensembl ID of the affected gene
`Variant Type`	The type of variant. `missense` for missense m
`Mutation`	The amnio acid change of this mutation
`Protein Position`	The protein position of the mutation
`Gene Name`	The Ensembl gene name of the affected gene
`HGVSc`	The HGVS coding sequence variant name
`HGVSp`	The HGVS protein sequence variant name
`HLA Allele`	The HLA allele for this prediction
`Peptide Length`	The peptide length of the epitope
`Sub-peptide Position`	The one-based position of the epitope within the
`Mutation Position`	The one-based position of the start of the mutati
`MT Epitope Seq`	The mutant epitope sequence
`WT Epitope Seq`	The wildtype (reference) epitope sequence at the
`Best MT Score Method`	Prediction algorithm with the lowest mutant ic50
`Best MT Score`	Lowest ic50 binding affinity of all prediction alg
`Corresponding WT Score`	ic50 binding affinity of the wildtype epitope. `NA`
`Corresponding Fold Change`	`Corresponding WT Score` / `Best MT S`
`Tumor DNA Depth`	Tumor DNA depth at this position. `NA` if VCF e
`Tumor DNA VAF`	Tumor DNA variant allele frequency (VAF) at th
`Tumor RNA Depth`	Tumor RNA depth at this position. `NA` if VCF e
`Tumor RNA VAF`	Tumor RNA variant allele frequency (VAF) at th
`Normal Depth`	Normal DNA depth at this position. `NA` if VCF
`Normal VAF`	Normal DNA variant allele frequency (VAF) at t
`Gene Expression`	Gene expression value for the annotated gene co
`Transcript Expression`	Transcript expression value for the annotated tra
`Median MT Score`	Median ic50 binding affinity of the mutant epito
`Median WT Score`	Median ic50 binding affinity of the wildtype epi
`Median Fold Change`	`Median WT Score` / `Median MT Score`.
`Individual Prediction Algorithm WT and MT Scores` (multiple)	ic50 scores for the `MT Epitope Seq` and `WT`
`cterm_7mer_gravy_score`	Mean hydropathy of last 7 residues on the C-ter
`max_7mer_gravy_score`	Max GRAVY score of any kmer in the amino ac
`difficult_n_terminal_residue` (T/F)	Is N-terminal amino acid a Glutamine, Glutamic
`c_terminal_cysteine` (T/F)	Is the C-terminal amino acid a Cysteine?
`c_terminal_proline` (T/F)	Is the C-terminal amino acid a Proline?
`cysteine_count`	Number of Cysteines in the amino acid sequence
`n_terminal_asparagine` (T/F)	Is the N-terminal amino acid a Asparagine?
`asparagine_proline_bond_count`	Number of Asparagine-Proline bonds. Problema
`Best Cleavage Position` (optional)	Position of the highest predicted cleavage score
`Best Cleavage Score` (optional)	Highest predicted cleavage score
`Cleavage Sites` (optional)	List of all cleavage positions and their cleavage
`Predicted Stability` (optional)	Stability of the pMHC-I complex
`Half Life` (optional)	Half-life of the pMHC-I complex
`Stability Rank` (optional)	The % rank stability of the pMHC-I complex

Table

Column Name	Description
`NetMHCstab allele` (optional)	Nearest neighbor to the `HLA Allele`. Used fo


17-mer peptide window
(length determined by the --peptide-sequence-length argument)
Sub-peptide position
pos 1
pos 2
pos 3
pos 4
pos 5
pos 6
pos 7
pos 8
pos 9
MT
Mutation position
9-mer epitope "registers"
(length determined by the --epitope-length argument)

1.5.2 filtered.condensed.ranked.tsv Report Columns

Column Name	Description
`Gene Name`	The Ensembl gene name of the affected gene.
`Mutation`	The amino acid change of this mutation.
`Protein Position`	The protein position of the mutation.
`HGVSc`	The HGVS coding sequence name.
`HGVSp`	The HGVS protein sequence name.
`HLA Allele`	The HLA allele for this prediction.
`Mutation Position`	The one-based position of the start of the mutation within the epitope sequence. `0` if the start of the mutation is before the epitope
`MT Epitope Seq`	Mutant epitope sequence.
`Median MT Score`	Median ic50 binding affinity of the mutant epitope across all prediction algorithms used
`Median WT Score`	Median ic50 binding affinity of the wildtype epitope across all prediction algorithms used. `NA` if there is no `WT Epitope Seq`.
`Median Fold Change`	`Median WT Score` / `Median MT Score`. `NA` if there is no `WT Epitope Seq`.
`Best MT Score`	Lowest ic50 binding affinity of all prediction algorithms used
`Corresponding WT Score`	ic50 binding affinity of the wildtype epitope. `NA` if there is no `WT Epitope Seq`.
`Corresponding Fold Change`	`Corresponding WT Score / Best MT Score`. `NA` if there is no `WT Epitope Seq`.
`Tumor DNA Depth`	Tumor DNA depth at this position. `NA` if VCF entry does not contain tumor DNA readcount annotation.
`Tumor DNA VAF`	Tumor DNA variant allele frequency at this position. `NA` if VCF entry does not contain tumor DNA readcount annotation.
`Tumor RNA Depth`	Tumor RNA depth at this position. `NA` if VCF entry does not contain tumor RNA readcount annotation.
`Tumor RNA VAF`	Tumor RNA variant allele frequency at this position. `NA` if VCF entry does not contain tumor RNA readcount annotation.
`Gene Expression`	Gene expression value at this position. `NA` if VCF entry does not contain gene expression annotation.
`Rank`	A priority rank for the neoepitope (best = 1).

The pVACseq Neoeptiope Priority Rank

Each of the following 4 criteria are assigned a rank-ordered value (worst = 1):

- B = Rank of the mutant IC50 binding affinity, with the lowest being the best. If the `--top-score-metric` is set to `median` (default) the `Median MT Score` is used. If it is set to `lowest` the `Best MT Score` is used.
- F = Rank of `Fold Change` between MT and WT alleles, with the highest being the best.
- M = Rank of mutant allele expression, calculated as (`Gene Expression` * `Tumor RNA VAF`), with the highest being the best.
- D = Rank of `Tumor DNA VAF`, with the highest being the best.

A score is calculated from the above ranks with the following formula: `B + F + (M * 2) + (D / 2)`. This score is then converted to a rank (best = 1).

Note: The pVACseq rank calculation detailed above is just one of many ways to prioritize neoeptiope candidates. The body of evidence in this area is still incomplete, and the methodology of ranking is likely to change substantially in future releases. While we have found this ranking useful, it is not a substitute for careful curation and validation efforts.

1.6 Filtering Commands

pVACseq currently offers four filters: a binding filter, a coverage filter, a transcript support level filter, and a top score filter.

These filters are always run automatically as part of the pVACseq pipeline using default cutoffs.

All filters can also be run manually on the filtered.tsv file to narrow the results down further, or they can be run on the all_epitopes.tsv file to apply different filtering thresholds.

The binding filter is used to remove neoantigen candidates that do not meet desired peptide:MHC binding criteria. The coverage filter is used to remove variants that do not meet desired read count and VAF criteria (in normal DNA and tumor DNA/RNA). The transcript support level filter is used to remove variant annotations based on low quality transcript annotations. The top score filter is used to select the most promising peptide candidate for each variant. Multiple candidate peptides from a single somatic variant can be caused by multiple peptide lengths, registers, HLA alleles, and transcript annotations.

Further details on each of these filters is provided below.

Note: The default values for filtering thresholds are suggestions only. While they are based on review of the literature and consultation with our clinical and immunology colleagues, your specific use case will determine the appropriate values.

1.6.1 Binding Filter

```
usage: pvacseq binding_filter [-h] [-b BINDING_THRESHOLD]
                              [-c MINIMUM_FOLD_CHANGE] [-m {lowest,median}]
                              [--exclude-NAs] [-a]
                              input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter.
  output_file           Output .tsv file containing list of filtered epitopes
                        based on binding affinity.

optional arguments:
  -h, --help            show this help message and exit
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  -c MINIMUM_FOLD_CHANGE, --minimum-fold-change MINIMUM_FOLD_CHANGE
```

(continues on next page)

(continued from previous page)

```
                        Minimum fold change between mutant binding score and
                        wild-type score. The default is 0, which filters no
                        results, but 1 is often a sensible option (requiring
                        that binding is better to the MT than WT). (default:
                        0)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the Best MT Score and corresponding Fold Change
                        (i.e. use the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the Median MT Score and Median
                        Fold Change (i.e. use the median MT ic50 binding score
                        and fold change of all chosen prediction methods).
                        (default: median)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
                        False)
  -a, --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
                        allele-specific binding thresholds run `pvacseq
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
```

The binding filter removes variants that don't pass the chosen binding threshold. The user can chose whether to apply this filter to the `lowest` or the `median` binding affinity score by setting the `--top-score-metric` flag. The `lowest` binding affinity score is recorded in the `Best MT Score` column and represents the lowest ic50 score of all prediction algorithms that were picked during the previous pVACseq run. The `median` binding affinity score is recorded in the `Median MT Score` column and corresponds to the median ic50 score of all prediction algorithms used to create the report. Be default, the binding filter runs on the `median` binding affinity.

The binding filter also offers the option to filter on `Fold Change` columns, which contain the ratio of the MT score to the WT Score. This option can be activated by setting the `--minimum-fold-change` threshold (to require that the mutant peptide is a better binder than the corresponding wild type peptide). If the `--top-score-metric` option is set to `lowest`, the `Corresponding Fold Change` column will be used (`Corresponding WT Score`/`Best MT Score`). If the `--top-score-metric` option is set to `median`, the `Median Fold Change` column will be used (`Median WT Score`/`Median MT Score`).

By default, entries with `NA` values will be included in the output. This behavior can be turned off by using the `--exclude-NAs` flag.

1.6.2 Coverage Filter

```
usage: pvacseq coverage_filter [-h] [--normal-cov NORMAL_COV]
                               [--tdna-cov TDNA_COV] [--trna-cov TRNA_COV]
                               [--normal-vaf NORMAL_VAF] [--tdna-vaf TDNA_VAF]
                               [--trna-vaf TRNA_VAF] [--expn-val EXPN_VAL]
                               [--exclude-NAs]
                               input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter
  output_file           Output .tsv file containing list of filtered epitopes
                        based on coverage and expression values
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  --normal-cov NORMAL_COV
                        Normal Coverage Cutoff. Sites above this cutoff will
                        be considered. (default: 5)
  --tdna-cov TDNA_COV   Tumor DNA Coverage Cutoff. Sites above this cutoff
                        will be considered. (default: 10)
  --trna-cov TRNA_COV   Tumor RNA Coverage Cutoff. Sites above this cutoff
                        will be considered. (default: 10)
  --normal-vaf NORMAL_VAF
                        Normal VAF Cutoff. Sites BELOW this cutoff in normal
                        will be considered. (default: 0.02)
  --tdna-vaf TDNA_VAF   Tumor DNA VAF Cutoff. Sites above this cutoff will be
                        considered. (default: 0.25)
  --trna-vaf TRNA_VAF   Tumor RNA VAF Cutoff. Sites above this cutoff will be
                        considered. (default: 0.25)
  --expn-val EXPN_VAL   Gene and Transcript Expression cutoff. Sites above
                        this cutoff will be considered. (default: 1.0)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
                        False)
```

If the input VCF contains readcount and/or expression annotations, then the coverage filter can be run again on the filtered.tsv report file to narrow down the results even further. You can also run this filter again on the all_epitopes.tsv report file to apply different cutoffs.

The general goals of these filters are to limit variants for neoepitope prediction to those with good read support and/or remove possible sub-clonal variants. In some cases the input VCF may have already been filtered in this fashion. This filter also allows for removal of variants that do not have sufficient evidence of RNA expression.

For more details on how to prepare input VCFs that contain all of these annotations, refer to the *Input File Preparation* section for more information.

By default, entries with `NA` values will be included in the output. This behavior can be turned off by using the `--exclude-NAs` flag.

1.6.3 Transcript Support Level Filter

```
usage: pvacseq transcript_support_level_filter [-h]
                                               [--maximum-transcript-support-level {1,
↪2,3,4,5}]
                                               [--exclude-NAs]
                                               input_file output_file

positional arguments:
  input_file            The all_epitopes.tsv or filtered.tsv pVACseq report
                        file to filter.
  output_file           Output .tsv file containting list of of filtered
                        epitopes based on transcript support level.

optional arguments:
  -h, --help            show this help message and exit
  --maximum-transcript-support-level {1,2,3,4,5}
                        The threshold to use for filtering epitopes on the
                        transcript support level. Keep all epitopes with a
                        transcript support level <= to this cutoff. (default:
```

(continues on next page)

(continued from previous page)

```
                        1)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
                        False)
```

This filter is used to eliminate variant annotations based on poorly-supported transcripts. By default, only transcripts with a transcript support level (TSL) of <=1 are kept. This threshold can be adjusted using the `--maximum-transcript-support-level` parameter.

By default, entries with `NA` values will be included in the output. This behavior can be turned off by using the `--exclude-NAs` flag.

1.6.4 Top Score Filter

```
usage: pvacseq top_score_filter [-h] [-m {lowest,median}]
                                input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter.
  output_file           Output .tsv file containing only the list of the top
                        epitope per variant.

optional arguments:
  -h, --help            show this help message and exit
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use for filtering. lowest:
                        Use the best MT Score (i.e. the lowest MT ic50 binding
                        score of all chosen prediction methods). median: Use
                        the median MT Score (i.e. the median MT ic50 binding
                        score of all chosen prediction methods). (default:
                        median)
```

This filter picks the top epitope for a variant. Epitopes with the same Chromosome - Start - Stop - Reference - Variant are identified as coming from the same variant.

In order to account for different splice sites among the transcripts of a variant that would lead to different peptides, this filter also takes into account the different transcripts returned by VEP and will return the top epitope for all transcripts if they are non-identical. If the resulting list of top epitopes for the transcripts of a variant is identical, the epitope for the transcript with the highest expression is returned. If this information is not available, the transcript with the lowest Ensembl ID is returned.

By default the `--top-score-metric` option is set to `median` which will apply this filter to the `Median MT Score` column and pick the epitope with the lowest median mutant ic50 score for each variant. If the `--top-score-metric` option is set to `lowest`, the `Best MT Score` column is instead used to make this determination.

It is important to note that there are several reasons why a particular variant can lead to multiple peptides with different predicted binding affinities. The following can result in multiple peptides and/or binding predictions for a single variant:

1. Different epitope lengths: specifying multiple epitope lengths results in similar but non-identical epitope sequences for each variant (e.g. KLPEPCPS, KLPEPCPST, KLPEPCPSTT, KLPEPCPSTTP). 2. Different registers: pVACseq will test epitopes where the mutation is in every position (e.g. EPCPSTTP, PEPCPSTT, LPEPCPST, KLPEPCPS, . . .). 3. Different transcripts: in some case the peptide sequence surrounding a variant will depend on the reference transcript sequence, particularly if there are alternative splice sites near the variant position. 4. Different HLA alleles: the HLA allele that produces the best predicted binding affinity is chosen. 5. A homozygous somatic variant with heterozygous proximal variants nearby may produce multiple different peptides.

The significance of choosing a single representative peptide can depend on your experimental or clinical aims. For example, if you are planning to use short peptide sequences exactly as they were assessed for binding affinity in pVACseq (e.g. specific 9-mers for in vitro experimental validation or perhaps a dendritic cell vaccine delivery approach) then the selection of a specific peptide from the possibilities caused by different lengths, registers, etc. is very important. In some cases you may wish to consider more criteria beyond which of these candidates has the best predicted binding affinity and gets chosen by the Top Score Filter.

On the other hand, if you plan to use synthetic long peptides (SLPs) or encode your candidates in a DNA vector, you will likely include flanking amino acids. This means that you often get a lot of the different short peptides that correspond to slightly different lengths or registers within the longer containing sequence. In this scenario, pVACseq's choice of a single candidate peptide by the Top Score Filter isn't actually that critical in the sense of losing other good candidates, because you may get them all anyway.

One important exception to this is the rare case where the same variant leads to different peptides in different transcripts (due to different splice site usage). If multiple transcripts are expressed and lead to distinct peptides, you may want to include both in your final list of candidates. The top score filter supports this case, as described above. This assumes you did not start with only a single transcript model for each gene (e.g. using the `--pick` option in VEP) and also that if you are requiring transcripts with TSL=1 that there are multiple qualifying transcripts that lead to different peptide sequences at the site of the variant. This will be fairly rare. Even though most genes have alternative transcripts, they often have only subtle differences in open reading frame and overall protein sequence, and only differences within the window that would influence a neoantigen candidate are consequential here.

1.7 Additional Commands

To make using pVACseq easier, several convenience methods are included in the package.

1.7.1 Download Example Data

```
usage: pvacseq download_example_data [-h] destination_directory

positional arguments:
  destination_directory
                        Directory for downloading example data

optional arguments:
  -h, --help            show this help message and exit
```

1.7.2 Install VEP Plugin

```
usage: pvacseq install_vep_plugin [-h] vep_plugins_path

positional arguments:
  vep_plugins_path  Path to your VEP_plugins directory

optional arguments:
  -h, --help        show this help message and exit
```

1.7.3 List Valid Alleles

```
usage: pvacseq valid_alleles [-h]
                             [-p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,
↪NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}]

optional arguments:
  -h, --help            show this help message and exit
  -p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,
↪NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}, --prediction-algorithm {MHCflurry,
↪MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,
↪SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use (default:
                        None)
```

1.7.4 List Allele-Specific Cutoffs

```
usage: pvacseq allele_specific_cutoffs [-h] [-a ALLELE]

optional arguments:
  -h, --help            show this help message and exit
  -a ALLELE, --allele ALLELE
                        The allele to use (default: None)
```

1.8 Optional Downstream Analysis Tools

1.8.1 Generate Protein Fasta

```
usage: pvacseq generate_protein_fasta [-h] [--input-tsv INPUT_TSV]
                                      [--mutant-only]
                                      [-d DOWNSTREAM_SEQUENCE_LENGTH]
                                      input_vcf peptide_sequence_length
                                      output_file

positional arguments:
  input_vcf             A VEP-annotated single-sample VCF containing
                        transcript, Wildtype protein sequence, and Downstream
                        protein sequence information.
  peptide_sequence_length
                        Length of the peptide sequence to use when creating
                        the FASTA.
  output_file           The output fasta file.

optional arguments:
  -h, --help            show this help message and exit
  --input-tsv INPUT_TSV
                        A pVACseq all_epitopes or filtered TSV file with
```

(continues on next page)

(continued from previous page)

```
                        epitopes to use for subsetting the input VCF to
                        peptides of interest. Only the peptide sequences for
                        the epitopes in the TSV will be used when creating the
                        FASTA. (default: None)
  --mutant-only         Only output mutant peptide sequences (default: False)
  -d DOWNSTREAM_SEQUENCE_LENGTH, --downstream-sequence-length DOWNSTREAM_SEQUENCE_
↪LENGTH
                        Cap to limit the downstream sequence length for
                        frameshifts when creating the fasta file. Use 'full'
                        to include the full downstream sequence. (default:
                        1000)
```

This tool will extract protein sequences surrounding supported protein altering variants in an input VCF file. One use case for this tool is to help select long peptides that contain short neoepitope candidates. For example, if pvacseq was run to predict nonamers (9-mers) that are good binders and the user wishes to select long peptide (e.g. 24-mer) sequences that contain the nonamer for synthesis or encoding in a DNA vector. The protein sequence extracted will correspond to the transcript sequence used in the annotated VCF. The alteration in the VCF (e.g. a somtic missense SNV) will be centered in the protein sequence returned (if possible). If the variant is near the beginning or end of the CDS, it will be as close to center as possible while returning the desired protein sequence length. If the variant causes a frameshift, the full downstream protein sequence will be returned unless the user specifies otherwise as described above.

1.8.2 Generate Condensed, Ranked Report

```
usage: pvacseq generate_condensed_ranked_report [-h] [-m {lowest,median}]
                                                input_file output_file

positional arguments:
  input_file            A pVACseq .all_epitopes.tsv or .filtered.tsv report
                        file
  output_file           The file path to write the condensed, ranked report
                        tsv to

optional arguments:
  -h, --help            show this help message and exit
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use for ranking epitopes by
                        binding-threshold and minimum fold change. lowest: Use
                        the best MT Score and Corresponding Fold Change (i.e.
                        the lowest MT ic50 binding score and corresponding
                        fold change of all chosen prediction methods). median:
                        Use the median MT Score and Median Fold Change (i.e.
                        the median MT ic50 binding score and fold change of
                        all chosen prediction methods). (default: median)
```

This tool will produce a condensed version of the filtered TSV with only the most important columns remaining, with a score for each neoepitope candidate added. Refer to the Output Files section for more details on the format of this report.

1.9 Common Errors

1.9.1 Input VCF Sample Information

VCF contains more than one sample but sample_name is not set.

pVACseq supports running with a multi-sample VCF as input. However, in this case it requires the user to pick the sample to analyze, as only variants that are called in the specified sample will be processed.

When running a multi-sample VCF the `sample_name` parameter is used to identify which sample to analyze. Take, for example, the following `#CHROM` VCF header:

```
#CHROM       POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  NORMAL ␣
↪TUMOR
```

This VCF contains two samples, `NORMAL` and `TUMOR`. Use `TUMOR` as the `sample_name` parameter to process the tumor sample, and `NORMAL` to process the normal sample.

If the input VCF only contains a single sample, the `sample_name` parameter does not need to match the sample name in the VCF.

sample_name not a sample ID in the #CHROM header of VCF

This error occurs when running a multi-sample VCF and the `sample_name` parameter doesn't match any of the sample IDs in the VCF `#CHROM` header. Take, for example, the following `#CHROM` header:

```
#CHROM       POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  NORMAL ␣
↪TUMOR
```

All columns after `FORMAT` are sample identifiers that can be used as the `sample_name` parameter when running pVACseq, depending on which sample the user wishes to process. Change the `sample_name` parameter of your `pvacseq run` command to match one of them.

normal_sample_name not a sample ID in the #CHROM header of VCF

Your `pvacseq run` command included the `--normal-sample-name` parameter. However, the argument chosen did not match any of the sample identifiers in the `#CHROM` header of the input VCF.

Take, for example, the following `#CHROM` VCF header:

```
#CHROM       POS     ID      REF     ALT     QUAL    FILTER  INFO    FORMAT  NORMAL ␣
↪TUMOR
```

All columns after `FORMAT` are sample identifiers that can be used as the `--normal-sample-name` parameter when running pVACseq, depending on which sample is the normal sample in the VCF. Change the `--normal-sample-name` parameter of your `pvacseq run` command to match the appropriate sample identifier.

VCF doesn't contain any sample genotype information.

pVACseq uses the sample genotype to identified which variants were called. Therefore, while a VCF without a `FORMAT` and sample column(s) is valid, it cannot be used in pVACseq. You will need to manually edit your VCF and add a `FORMAT` and sample column with the `GT` genotype field. For more information on this formatting please see the VCF specification for your specific VCF version.

1.9.2 Input VCF Compression and Indexing

Input VCF needs to be bgzipped when running with a proximal variants VCF.

When running pVACseq with the `--proximal-variants-vcf` argument, the main input VCF needs to be bgzipped and tabix indexed. See *the Input File Preparation section* of the documentation for instructions on how to do so.

Proximal variants VCF needs to be bgzipped.

The VCF provided via the `--proximal-variants-vcf` argument needs to be bgzipped and tabix indexed. See *the Input File Preparation section* of the documentation for instructions on how to do so.

No .tbi file found for input VCF. Input VCF needs to be tabix indexed if processing with proximal variants.

When running pVACseq with the `--proximal-variants-vcf` argument, the main input VCF needs to be bgzipped and tabix indexed. See *the Input File Preparation section* of the documentation for instructions on how to do so.

No .tbi file found for proximal variants VCF. Proximal variants VCF needs to be tabix indexed.

The VCF provided via the `--proximal-variants-vcf` argument needs to be bgzipped and tabix indexed. See *the Input File Preparation section* of the documentation for instructions on how to do so.

1.9.3 Input VCF VEP Annotation

Input VCF does not contain a CSQ header. Please annotate the VCF with VEP before running it.

pVACseq requires the input VCF to be annotated by VEP. The provided input VCF doesn't contain a `CSQ INFO` header. This indicates that it has not been annotated. *The Input File Preparation section* of the documentation provides instructions on how to annotate your VCF with VEP.

VCF doesn't contain VEP DownstreamProtein annotations. Please re-annotate the VCF with VEP and the Wildtype and Downstream plugins.

Although the input VCF was annotated with VEP, it is missing the required annotations provided by the VEP Downstream plugin. The input VCF will need to be reannotated using all of the required arguments as outlined in the *Input File Preparation section* of the documentation.

VCF doesn't contain VEP WildtypeProtein annotations. Please re-annotate the VCF with VEP and the Wildtype and Downstream plugins.

Although the input VCF was annotated with VEP, it is missing the required annotations provided by the VEP Wildtype plugin. The input VCF will need to be reannotated using all of the required arguments as outlined in the *Input File Preparation section* of the documentation.

Proximal Variants VCF does not contain a CSQ header. Please annotate the VCF with VEP before running it.

When running pVACseq with the `--proximal-variants-vcf` argument, that proximal variants VCF needs to be annotated by VEP. The provided proximal variants VCF doesn't contain a `CSQ INFO` header. This indicates that it has not been annotated. *The Input File Preparation section* of the documentation provides instructions on how to annotate your VCF with VEP.

There was a mismatch between the actual wildtype amino acid sequence and the expected amino acid sequence. Did you use the same reference build version for VEP that you used for creating the VCF?

This error occurs when the reference nucleotide at a specific position is different than the Ensembl transcript nucleotide at the same position. This results in the mutant amino acid in the `Amino_acids` VEP annotation being different from the amino acid of the transcript protein sequence as predicted by the Wildtype plugin. The `Amino_acids` VEP annotation is based on the reference and alternate nucleotides of the variant while the `WildtypeProtein` prediction is based on the Ensembl transcript nucleotide sequence.

This points to a fundamental disagreement between the reference that was used during alignment and variant calling and the Ensembl reference. This mismatch cannot be resolved by pVACseq, which is why this error is fatal.

Here are a few things that might resolve this error:

- Checking that the build of the VEP cache matches the alignment build and downloading the correct cache if there is a build mismatch (such as a build 38 cache with a build 37 VCF, or vice versa)
- Using the `--assembly` parameter during VEP annotation with the correct build version to match your VCF
- Using the `fasta` parameter during VEP annotation with the reference used to create the VCF
- Manually fixing the reference bases in your VCF to match the one expected by Ensembl
- Realigning and redoing variant calling on your sample with a reference that matches what is expected by VEP

If this mismatch cannot be resolved the VCF cannot be used by pVACseq.

1.9.4 Other

The TSV file is empty. Please check that the input VCF contains missense, inframe indel, or frameshift mutations.

None of the variants in the VCF file are supported by pVACseq.

Illegal instruction (core dumped)

This issue may occur when you are trying to run the tensorflow-based prediction algorithms MHCnuggets and/or MHCflurry. This indicates that your computer's hardware does not support the version of tensorflow that is installed. Downgrading tensorflow manually to version 1.5.0 (`pip install tensorflow==1.5.0`) should solve this problem.

1.10 Frequently Asked Questions

What type of variants does pVACseq support?

pVACseq makes predictions for all transcripts of a variant that were annotated as `missense_variant`, `inframe_insertion`, `inframe_deletion`, inframe `protein_altering_variant`, or `frameshift_variant` by VEP as long as the transcript was not also annotated as `start_lost`. In addition, pVACseq only includes variants that were called as homozygous or heterozygous variant. Variants that were not called in the sample specified are skipped (determined by examining the `GT` genotype field in the VCF).

My pVACseq command has been running for a long time. Why is that?

The rate-limiting factor in running pVACseq is the number of calls that are made to the IEDB software for binding score predictions.

Note: It is generally faster to make IEDB calls using a local install of IEDB than using the IEDB web API. It is, therefore, recommended to use a local IEDB install for any in-depth analysis. You should either install IEDB locally yourself or use the pvactools docker image that includes it.

There are a number of factors that determine the number of IEDB calls to be made:

- Number of variants in your VCF

  pVACseq will make predictions for each missense, inframe insertion, inframe deletion, protein altering, and frameshift variant in your VCF.

  Speedup suggestion: Split the VCF into smaller subsets and process each one individually, in parallel.

- Number of transcripts for each variant

  pVACseq will make predictions for each transcript of a supported variant individually. The number of transcripts for each variant depends on how VEP was run when the VCF was annotated.

  Speedup suggestion: Use the `--pick` option when running VEP to annotate each variant with the top transcript only.

- The `--fasta-size` parameter value

  pVACseq takes an input VCF and creates a wildtype and a mutant FASTA for each transcript. The number of FASTA entries that get submitted to IEDB at a time is limited by the `--fasta-size` parameter in order to reduce the load on the IEDB servers. The smaller the FASTA size, the more calls have to be made to IEDB.

  Speedup suggestion: When using a local IEDB install, increase the size of this parameter.

- Number of prediction algorithms, epitope lengths, and HLA-alleles

  One call to IEDB is made for each combination of these parameters for each chunk of FASTA sequences. That means, for example, when 8 prediction algorithms, 4 epitope lengths (8-11), and 6 HLA-alleles are chosen, 7*4*6=192 calls to IEDB have to be made for each chunk of FASTA.

  Speedup suggestion: Reduce the number of prediction algorithms, epitope lengths, and/or HLA-alleles to the ones that will be the most meaningful for your analysis. For example, the NetMHCcons method is already a consensus method between NetMHC, NetMHCpan, and PickPocket. If NetMHCcons is chosen, you may want to omit the underlying prediction methods. Likewise, if you want to run NetMHC, NetMHCpan, and PickPocket individually, you may want to skip NetMHCcons.

- `--downstream-sequence-length` parameter value

  This parameter determines how many amino acids of the downstream sequence after a frameshift mutation will be included in the wildtype FASTA sequence. The shorter the downstream sequence length, the lower the number of epitopes that IEDB needs to make binding predictions for.

  Speedup suggestion: Reduce the value of this parameter.

- `-t` parameter value

  This parameter determines the number of threads pvacseq will use for parallel processing.

  Speedup suggestion: Use a host with multiple cores and sufficient memory and use a larger number of threads.

My pVACseq output file does not contain entries for all of the alleles I chose. Why is that?

There could be a few reasons why the pVACseq output does not contain predictions for alleles:

- The alleles you picked might have not been compatible with the prediction algorithm and/or epitope lengths chosen. In that case no calls for that allele would've been made and a status message would've printed to the screen.
- It could be that all epitope predictions for some alleles got filtered out. You can check the `<sample_name>.all_epitopes.tsv` file to see all called epitopes before filtering.

Why are some values in the WT Epitope Seq column NA ?

Not all mutant epitope sequences will have a corresponding wildtype epitope sequence. This occurs when the mutant epitope sequence is novel and a comparison is therefore not meaningful. For example:

- An epitope in the downstream portion of a frameshift might not have a corresponding wildtype epitope at the same position at all. The epitope is completely novel.
- An epitope that overlaps an inframe indel or multinucleotide polymorphism (MNP) might have a large number of amino acids that are different from the wildtype epitope at the corresponding position. If less than half of

the amino acids between the mutant epitope sequence and the corresponding wildtype sequence match, the corresponding wildtype sequence in the report is set to `NA`.

What filters are applied during a pVACseq run?

By default we filter the neoepitopes on their binding score. If readcount and/or expression annotations are available in the VCF we also filter on the depth, VAF, and gene/trancript FPKM. In addition, candidates where the mutant epitope sequence is the same as the wildtype epitope sequence will also be filtered out.

How can I see all of the candidate epitopes without any filters applied?

The `<sample_name>.all_epitopes.tsv` will contain all of the epitopes predicted before filters are applied.

Why have some of my epitopes been filtered out even though the Best MT Score is below 500?

By default, the binding filter will be applied to the `Median MT Score` column. This is the median score value among all chosen prediction algorithms. The `Best MT Score` column shows the lowest score among all chosen prediction algorithms. To change this behavior and apply the binding filter to the `Best MT Score` column you may set the `--top-score-metric` parameter to `lowest`.

Why are entries with NA in the VAF and depth columns not filtered?

We do not filter out `NA` entries for depth and VAF since there is not enough information to determine whether the cutoff has been met one way or another.

Why do some of my epitopes have no score predictions for certain prediction methods?

Not all prediction methods support all epitope lengths or all alleles. To see a list of supported alleles for a prediction method you may use the `pvacseq valid_alleles` *command*. For more details on each algorithm refer to the IEDB MHC Class I and Class II documentation.

How is pVACseq licensed?

pVACseq is licensed under the open source license NPOSL-3.0. If you would like to discuss a license for commercial applications, please contact us.

How do I cite pVACseq?

Jasreet Hundal+, Susanna Kiwala+, Joshua McMichael, Christopher A Miller, Alexander T Wollam, Huiming Xia, Connor J Liu, Sidi Zhao, Yang-Yang Feng, Aaron P Graubert, Amber Z Wollam, Jonas Neichin, Megan Neveau, Jason Walker, William E Gillanders, Elaine R Mardis, Obi L Griffith, Malachi Griffith. pVACtools: a computational toolkit to select and visualize cancer neoantigens. (+)equal contribution. bioRxiv 501817; doi: https://doi.org/10.1101/501817

Jasreet Hundal, Susanna Kiwala, Yang-Yang Feng, Connor J. Liu, Ramaswamy Govindan, William C. Chapman, Ravindra Uppaluri, S. Joshua Swamidass, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. Accounting for proximal variants improves neoantigen prediction. Nature Genetics. 2018, DOI: 10.1038/s41588-018-0283-9. PMID: 30510237.

Jasreet Hundal, Beatriz M. Carreno, Allegra A. Petti, Gerald P. Linette, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. pVACseq: A genome-guided in silico approach to identifying tumor neoantigens. Genome Medicine. 2016, 8:11, DOI: 10.1186/s13073-016-0264-5. PMID: 26825632.

pVACbind

CHAPTER 2

pVACbind

This component of the pVACtools is used to predict neoantigens for the peptides in a FASTA file.

2.1 Prerequisites

The input to pVACbind is a FASTA file of peptide sequences.

2.2 Getting Started

pVACbind provides a set of example data to show the expected format of input and output files. You can download the data set by running the `pvacbind download_example_data` *command*.

The example data output can be reproduced by running the following command:

```
pvacbind run \
<example_data_dir>/input.fasta \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
<output_dir> \
-e 8,9,10
```

A detailed description of all command options can be found on the *Usage* page.

2.3 Usage

Warning: Using a local IEDB installation is strongly recommended for larger datasets or when the making predictions for many alleles, epitope lengths, or prediction algorithms. More information on how to install IEDB locally can be found on the *Installation* page.

```
usage: pvacbind run [-h] [-e EPITOPE_LENGTH]
                    [--iedb-install-directory IEDB_INSTALL_DIRECTORY]
                    [-b BINDING_THRESHOLD]
                    [--allele-specific-binding-thresholds]
                    [-m {lowest,median}] [-r IEDB_RETRIES] [-k] [-t N_THREADS]
                    [--net-chop-method {cterm,20s}] [--netmhc-stab]
                    [--net-chop-threshold NET_CHOP_THRESHOLD]
                    [-a {sample_name}] [-s FASTA_SIZE] [--exclude-NAs]
                    input_file sample_name allele
                    {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}
                    [{MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign} ...]
                    output_dir

positional arguments:
  input_file            A FASTA file
  sample_name           The name of the sample being processed. This will be
                        used as a prefix for output files.
  allele                Name of the allele to use for epitope prediction.
                        Multiple alleles can be specified using a comma-
                        separated list. For a list of available alleles, use:
                        `pvacseq valid_alleles`.
  {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,
↪PickPocket,SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use. Multiple
                        prediction algorithms can be specified, separated by
                        spaces.
  output_dir            The directory for writing all result files.

optional arguments:
  -h, --help            show this help message and exit
  -e EPITOPE_LENGTH, --epitope-length EPITOPE_LENGTH
                        Length of subpeptides (neoepitopes) to predict.
                        Multiple epitope lengths can be specified using a
                        comma-separated list. Typical epitope lengths vary
                        between 8-11. Required for Class I prediction
                        algorithms. (default: None)
  --iedb-install-directory IEDB_INSTALL_DIRECTORY
                        Directory that contains the local installation of IEDB
```

(continues on next page)

(continued from previous page)

```
                        MHC I and/or MHC II. (default: None)
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
                        allele-specific binding thresholds run `pvacbind
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the best MT Score and Corresponding Fold Change
                        (i.e. the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the median MT Score and Median
                        Fold Change (i.e. the median MT ic50 binding score and
                        fold change of all chosen prediction methods).
                        (default: median)
  -r IEDB_RETRIES, --iedb-retries IEDB_RETRIES
                        Number of retries when making requests to the IEDB
                        RESTful web interface. Must be less than or equal to
                        100. (default: 5)
  -k, --keep-tmp-files  Keep intermediate output files. This might be useful
                        for debugging purposes. (default: False)
  -t N_THREADS, --n-threads N_THREADS
                        Number of threads to use for parallelizing peptide-MHC
                        binding prediction calls. (default: 1)
  --net-chop-method {cterm,20s}
                        NetChop prediction method to use ("cterm" for C term
                        3.0, "20s" for 20S 3.0). C-term 3.0 is trained with
                        publicly available MHC class I ligands and the authors
                        believe that is performs best in predicting the
                        boundaries of CTL epitopes. 20S is trained with in
                        vitro degradation data. (default: None)
  --netmhc-stab         Run NetMHCStabPan after all filtering and add
                        stability predictions to predicted epitopes. (default:
                        False)
  --net-chop-threshold NET_CHOP_THRESHOLD
                        NetChop prediction threshold (increasing the threshold
                        results in better specificity, but worse sensitivity).
                        (default: 0.5)
  -a {sample_name}, --additional-report-columns {sample_name}
                        Additional columns to output in the final report. If
                        sample_name is chosen, this will add a column with the
                        sample name in every row of the output. This can be
                        useful if you later want to concatenate results from
                        multiple individuals into a single file. (default:
                        None)
  -s FASTA_SIZE, --fasta-size FASTA_SIZE
                        Number of FASTA entries per IEDB request. For some
                        resource-intensive prediction algorithms like
                        Pickpocket and NetMHCpan it might be helpful to reduce
                        this number. Needs to be an even number. (default:
                        200)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
```

(continues on next page)

(continued from previous page)

```
                              False)
```

2.4 Output Files

The pVACbind pipeline will write its results in separate folders depending on which prediction algorithms were chosen:

- `MHC_Class_I`: for MHC class I prediction algorithms
- `MHC_Class_II`: for MHC class II prediction algorithms
- `combined`: If both MHC class I and MHC class II prediction algorithms were run, this folder combines the neoeptiope predictions from both

Each folder will contain the same list of output files (listed in the order created):

File Name	Description
`<sample_name>.tsv`	An intermediate file with variant information parsed from the input files.
`<sample_name>.tsv_<chunks>` (multiple)	The above file but split into smaller chunks for easier processing with IEDB.
`<sample_name>.all_epitopes.tsv`	A list of all predicted epitopes and their binding affinity scores, with additional variant information from the `<sample_name>.tsv`.
`<sample_name>.filtered.tsv`	The above file after applying all filters, with cleavage site and stability predictions added.

2.4.1 all_epitopes.tsv and filtered.tsv Report Columns

Column Name	Description
`Mutation`	The FASTA ID of the peptide sequence the epitope belongs to
`HLA Allele`	The HLA allele for this prediction
`Sub-peptide Position`	The one-based position of the epitope in the protein sequence used to make the prediction
`Epitope Seq`	The epitope sequence
`Median Score`	Median ic50 binding affinity of the epitope of all prediction algorithms used
`Best Score`	Lowest ic50 binding affinity of all prediction algorithms used
`Best Score Method`	Prediction algorithm with the lowest ic50 binding affinity for this epitope
`Individual Prediction Algorithm Scores` (multiple)	ic50 scores for the `Epitope Seq` for the individual prediction algorithms used
`cterm_7mer_gravy_score`	Mean hydropathy of last 7 residues on the C-terminus of the peptide
`max_7mer_gravy_score`	Max GRAVY score of any kmer in the amino acid sequence. Used to determine if there are any extremely hydrophobic regions within a longer amino acid sequence.
`difficult_n_terminal_residue` (T/F)	Is N-terminal amino acid a Glutamine, Glutamic acid, or Cysteine?
`c_terminal_cysteine` (T/F)	Is the C-terminal amino acid a Cysteine?
`c_terminal_proline` (T/F)	Is the C-terminal amino acid a Proline?
`cysteine_count`	Number of Cysteines in the amino acid sequence. Problematic because they can form disulfide bonds across distant parts of the peptide
`n_terminal_asparagine` (T/F)	Is the N-terminal amino acid a Asparagine?
`asparagine_proline_bond_count`	Number of Asparagine-Proline bonds. Problematic because they can spontaneously cleave the peptide
`Best Cleavage Position` (optional)	Position of the highest predicted cleavage score
`Best Cleavage Score` (optional)	Highest predicted cleavage score
`Cleavage Sites` (optional)	List of all cleavage positions and their cleavage score
`Predicted Stability` (optional)	Stability of the pMHC-I complex
`Half Life` (optional)	Half-life of the pMHC-I complex
`Stability Rank` (optional)	The % rank stability of the pMHC-I complex
`NetMHCstab allele` (optional)	Nearest neighbor to the `HLA Allele`. Used for NetMHCstab prediction

2.5 Filtering Commands

pVACbind currently offers two filters: a binding filter and a top score filter.

These filters are always run automatically as part of the pVACbind pipeline using default cutoffs.

All filters can also be run manually on the filtered.tsv file to narrow the results down further, or they can be run on the all_epitopes.tsv file to apply different filtering thresholds.

The binding filter is used to remove neoantigen candidates that do not meet desired peptide:MHC binding criteria. The top score filter is used to select the most promising peptide candidate for each variant. Multiple candidate peptides from a single somatic variant can be caused by multiple peptide lengths, registers, HLA alleles, and transcript annotations.

Further details on each of these filters is provided below.

Note: The default values for filtering thresholds are suggestions only. While they are based on review of the literature and consultation with our clinical and immunology colleagues, your specific use case will determine the appropriate values.

2.5.1 Binding Filter

```
usage: pvacbind binding_filter [-h] [-b BINDING_THRESHOLD]
                               [-m {lowest,median}] [--exclude-NAs] [-a]
                               input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter.
  output_file           Output .tsv file containing list of filtered epitopes
                        based on binding affinity.

optional arguments:
  -h, --help            show this help message and exit
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the Best MT Score and corresponding Fold Change
                        (i.e. use the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the Median MT Score and Median
                        Fold Change (i.e. use the median MT ic50 binding score
                        and fold change of all chosen prediction methods).
                        (default: median)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
                        False)
  -a, --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
                        allele-specific binding thresholds run `pvacbind
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
```

The binding filter removes variants that don't pass the chosen binding threshold. The user can chose whether to apply this filter to the `lowest` or the `median` binding affinity score by setting the `--top-score-metric` flag. The `lowest` binding affinity score is recorded in the `Best MT Score` column and represents the lowest ic50 score of all prediction algorithms that were picked during the previous pVACseq run. The `median` binding affinity score is recorded in the `Median MT Score` column and corresponds to the median ic50 score of all prediction algorithms used to create the report. Be default, the binding filter runs on the `median` binding affinity.

By default, entries with `NA` values will be included in the output. This behavior can be turned off by using the `--exclude-NAs` flag.

2.5.2 Top Score Filter

```
usage: pvacbind top_score_filter [-h] [-m {lowest,median}]
                                 input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter.
  output_file           Output .tsv file containing only the list of the top
                        epitope per variant.

optional arguments:
  -h, --help            show this help message and exit
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use for filtering. lowest:
                        Use the best MT Score (i.e. the lowest MT ic50 binding
                        score of all chosen prediction methods). median: Use
                        the median MT Score (i.e. the median MT ic50 binding
                        score of all chosen prediction methods). (default:
                        median)
```

This filter picks the top epitope for a variant. By default the `--top-score-metric` option is set to `median` which will apply this filter to the `Median MT Score` column and pick the epitope with the lowest median mutant ic50 score for each variant. If the `--top-score-metric` option is set to `lowest`, the `Best MT Score` column is instead used to make this determination.

2.6 Additional Commands

To make using pVACbind easier, several convenience methods are included in the package.

2.6.1 Download Example Data

```
usage: pvacbind download_example_data [-h] destination_directory

positional arguments:
  destination_directory
                        Directory for downloading example data

optional arguments:
  -h, --help            show this help message and exit
```

2.6.2 List Valid Alleles

```
usage: pvacbind valid_alleles [-h]
                              [-p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,
↪NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}]

optional arguments:
  -h, --help            show this help message and exit
  -p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,
↪NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}, --prediction-algorithm {MHCflurry,
↪MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,
↪SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use (default:
                        None)
```

2.6.3 List Allele-Specific Cutoffs

```
usage: pvacbind allele_specific_cutoffs [-h] [-a ALLELE]

optional arguments:
  -h, --help            show this help message and exit
  -a ALLELE, --allele ALLELE
                        The allele to use (default: None)
```

CHAPTER 3

pVACfuse

This component of the pVACtools workflow provides support for predicting neoantigens from gene fusions. Currently, fusion variants as reported by INTEGRATE-Neo are supported, and any of the binding affinity prediction software available in pVACseq can be used for binding prediction.

3.1 Prerequisites

3.1.1 Fusion detection and annotation

pVACfuse accepts two types of inputs, either an annotated bedpe file with fusion information from INTEGRATE-Neo or a output directory from AGFusion (recommended).

AGFusion

AGFusion allows a user to annotate output files from several fusion callers using the `agfusion batch` command. The below example is for annotating the output from the STAR-Fusion caller but many other fusion callers are supported. For a full list see the AGFusion documentation.

```
agfusion batch \
-f <star_fusion_tsv> \
-a starfusion \
-db agfusion.homo_sapiens.87.db \
- <output_directory> \
--middlestar \
--noncanonical
```

The `--middlestar` flag is required in order to use the ouput with pVACfuse. This will indicate the fusion position in the fusion peptide sequence.

The `--noncanonical` flag is optional and can be used to annotate the fusion with informations from all possible transcripts. By default only canonical transcripts are used.

INTEGRATE-Neo

Fusion detection will be preformed using INTEGRATE with annotations from INTEGRATE-Neo. It should be possible to start with fusions from another caller, convert the output to bedpe format, annotate the bedpe with INTEGRATE-Neo and then feed these candidates into pVACfuse.

1. Align RNA with Tophat2 (a requirement of INTEGRATE) to obtain accepted_hits.bam and unmapped.bam
2. (OPTIONAL) Align WGS DNA with BWA aln/sampe (NOT MEM, a requirement of INTEGRATE) to obtain tumor.dna.bam and normal.dna.bam
3. Produce a gene annotations file with gtfToGenePred

```
gtfToGenePred -genePredExt -geneNameAsName2 ref.gtf ref.genePred
cut -f 1-10,12 ref.genePred > tmp.txt
echo -e "#GRCh37.ensGene.name\tGRCh37.ensGene.chrom\tGRCh37.ensGene.strand\tGRCh37.
↪ensGene.txStart\tGRCh37.ensGene.txEnd\tGRCh37.ensGene.cdsStart\tGRCh37.ensGene.
↪cdsEnd\tGRCh37.ensGene.exonCount\tGRCh37.ensGene.exonStarts\tGRCh37.ensGene.
↪exonEnds\tGRCh37.ensemblToGeneName.value" > annot.txt
cat tmp.txt >> annot.txt
```

4. Run INTEGRATE to obtain fusions.bedpe

```
Integrate fusion ref.fa annot.txt bwts accepted_hits.bam unmappeds.bam [tumor.dna.bam␣
↪normal.dna.bam | tumor.dna.bam]
```

5. Run INTEGRATE-Neo to obtain annotated fusions bedpe file

```
integrate-neo.py -t hla.optitype -f fusions.bedpe -r ref.fa -g ref.genePred -k
```

3.2 Getting Started

pVACfuse provides a set of example data to show the expected format of input and output files. You can download the data set by running the `pvacfuse download_example_data` *command*.

There are two option as to how to run pVACfuse. It accepts either a INTEGRATE-neo output bedpe file or a AGFusion output directory.

The following command is an example for how to run pVACfuse with an INTEGRATE-neo bedpe file and will regenerate the `results_from_integrate_neo` example data:

```
pvacfuse run \
<example_data_dir>/fusions.bedpe.annot \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
```

(continues on next page)

(continued from previous page)

```
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
<output_dir> \
-e 8,9,10
```

The `results_from_agfusion` example data can be regenerated like so:

```
pvacfuse run \
<example_data_dir>/agfusion/ \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
<output_dir> \
-e 8,9,10
```

A detailed description of all command options can be found on the *Usage* page.

3.3 Usage

Warning: Using a local IEDB installation is strongly recommended for larger datasets or when the making predictions for many alleles, epitope lengths, or prediction algorithms. More information on how to install IEDB locally can be found on the *Installation* page.

```
usage: pvacfuse run [-h] [-e EPITOPE_LENGTH]
                    [--iedb-install-directory IEDB_INSTALL_DIRECTORY]
                    [-b BINDING_THRESHOLD]
                    [--allele-specific-binding-thresholds]
                    [-m {lowest,median}] [-r IEDB_RETRIES] [-k] [-t N_THREADS]
                    [--net-chop-method {cterm,20s}] [--netmhc-stab]
                    [--net-chop-threshold NET_CHOP_THRESHOLD]
                    [-a {sample_name}] [-s FASTA_SIZE] [--exclude-NAs]
                    [-l PEPTIDE_SEQUENCE_LENGTH]
                    [-d DOWNSTREAM_SEQUENCE_LENGTH]
                    input_file sample_name allele
                    {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}
                    [{MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign} ...]
                    output_dir

positional arguments:
  input_file            An INTEGRATE-Neo annotated bedpe file with fusions or
                        a AGfusion output directory.
  sample_name           The name of the sample being processed. This will be
                        used as a prefix for output files.
  allele                Name of the allele to use for epitope prediction.
                        Multiple alleles can be specified using a comma-
                        separated list. For a list of available alleles, use:
```

(continues on next page)

(continued from previous page)

```
                        `pvacseq valid_alleles`.
  {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,
↪PickPocket,SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use. Multiple
                        prediction algorithms can be specified, separated by
                        spaces.
  output_dir            The directory for writing all result files.

optional arguments:
  -h, --help            show this help message and exit
  -e EPITOPE_LENGTH, --epitope-length EPITOPE_LENGTH
                        Length of subpeptides (neoepitopes) to predict.
                        Multiple epitope lengths can be specified using a
                        comma-separated list. Typical epitope lengths vary
                        between 8-11. Required for Class I prediction
                        algorithms. (default: None)
  --iedb-install-directory IEDB_INSTALL_DIRECTORY
                        Directory that contains the local installation of IEDB
                        MHC I and/or MHC II. (default: None)
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
                        allele-specific binding thresholds run `pvacfuse
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the best MT Score and Corresponding Fold Change
                        (i.e. the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the median MT Score and Median
                        Fold Change (i.e. the median MT ic50 binding score and
                        fold change of all chosen prediction methods).
                        (default: median)
  -r IEDB_RETRIES, --iedb-retries IEDB_RETRIES
                        Number of retries when making requests to the IEDB
                        RESTful web interface. Must be less than or equal to
                        100. (default: 5)
  -k, --keep-tmp-files  Keep intermediate output files. This might be useful
                        for debugging purposes. (default: False)
  -t N_THREADS, --n-threads N_THREADS
                        Number of threads to use for parallelizing peptide-MHC
                        binding prediction calls. (default: 1)
  --net-chop-method {cterm,20s}
                        NetChop prediction method to use ("cterm" for C term
                        3.0, "20s" for 20S 3.0). C-term 3.0 is trained with
                        publicly available MHC class I ligands and the authors
                        believe that is performs best in predicting the
                        boundaries of CTL epitopes. 20S is trained with in
                        vitro degradation data. (default: None)
  --netmhc-stab         Run NetMHCStabPan after all filtering and add
                        stability predictions to predicted epitopes. (default:
                        False)
```

(continues on next page)

(continued from previous page)

```
  --net-chop-threshold NET_CHOP_THRESHOLD
                        NetChop prediction threshold (increasing the threshold
                        results in better specificity, but worse sensitivity).
                        (default: 0.5)
  -a {sample_name}, --additional-report-columns {sample_name}
                        Additional columns to output in the final report. If
                        sample_name is chosen, this will add a column with the
                        sample name in every row of the output. This can be
                        useful if you later want to concatenate results from
                        multiple individuals into a single file. (default:
                        None)
  -s FASTA_SIZE, --fasta-size FASTA_SIZE
                        Number of FASTA entries per IEDB request. For some
                        resource-intensive prediction algorithms like
                        Pickpocket and NetMHCpan it might be helpful to reduce
                        this number. Needs to be an even number. (default:
                        200)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
                        False)
  -l PEPTIDE_SEQUENCE_LENGTH, --peptide-sequence-length PEPTIDE_SEQUENCE_LENGTH
                        Length of the peptide sequence to use when creating
                        the FASTA. (default: 21)
  -d DOWNSTREAM_SEQUENCE_LENGTH, --downstream-sequence-length DOWNSTREAM_SEQUENCE_
↪LENGTH
                        Cap to limit the downstream sequence length for
                        frameshifts when creating the FASTA file. Use 'full'
                        to include the full downstream sequence. (default:
                        1000)
```

3.4 Output Files

The pVACfuse pipeline will write its results in separate folders depending on which prediction algorithms were chosen:

- `MHC_Class_I`: for MHC class I prediction algorithms
- `MHC_Class_II`: for MHC class II prediction algorithms
- `combined`: If both MHC class I and MHC class II prediction algorithms were run, this folder combines the neoeptiope predictions from both

Each folder will contain the same list of output files (listed in the order created):

File Name	Description
`<sample_name>.tsv`	An intermediate file with variant and transcript information parsed from the input file(s).
`<sample_name>.tsv_<chunks>` (multiple)	The above file but split into smaller chunks for easier processing with IEDB.
`<sample_name>.all_epitopes.tsv`	A list of all predicted epitopes and their binding affinity scores, with additional variant information from the `<sample_name>.tsv`.
`<sample_name>.filtered.tsv`	The above file after applying all filters, with cleavage site and stability predictions added.
`<sample_name>.filtered.condensed.ranked.tsv`	A condensed version of the filtered TSV with only the most important columns remaining, with a priority score for each neoepitope candidate added.

3.4.1 all_epitopes.tsv and filtered.tsv Report Columns

In order to keep the outputs consistent, pVACfuse uses the same output columns as pVACseq but some of the values will be `NA` if a column doesn't apply to pVACfuse.

Column Name	Description
`Chromosome`	The chromosome of the 5p and 3p portion of the
`Start`	The start position of the 5p and 3p portion of the
`Stop`	The stop position of the 5p and 3p portion of the
`Reference`	`fusion`
`Variant`	`fusion`
`Transcript`	The Ensembl IDs of the affected transcripts
`Transcript Support Level`	`NA`
`Ensembl Gene ID`	`NA`
`Variant Type`	The type of fusion. `inframe_fusion` for infr
`Mutation`	`NA`
`Protein Position`	The position of the fusion in the fusion protein s
`Gene Name`	The Ensembl gene names of the affected genes
`HGVSc`	`NA`
`HGVSp`	`NA`
`HLA Allele`	The HLA allele for this prediction
`Peptide Length`	The peptide length of the epitope
`Sub-peptide Position`	The one-based position of the epitope in the prot
`Mutation Position`	`NA`
`MT Epitope Seq`	Mutant epitope sequence
`WT Epitope Seq`	`NA`
`Best MT Score Method`	Prediction algorithm with the lowest mutant ic50
`Best MT Score`	Lowest ic50 binding affinity of all prediction alg
`Corresponding WT Score`	`NA`
`Corresponding Fold Change`	`NA`
`Tumor DNA Depth`	`NA`
`Tumor DNA VAF`	`NA`
`Tumor RNA Depth`	`NA`
`Tumor RNA VAF`	`NA`
`Normal Depth`	`NA`
`Normal VAF`	`NA`
`Gene Expression`	`NA`

Table 1 – continued from pr

Column Name	Description
`Transcript Expression`	NA
`Median MT Score`	Median ic50 binding affinity of the mutant epito
`Median WT Score`	NA
`Median Fold Change`	NA
`Individual Prediction Algorithm WT and MT Scores` (multiple)	ic50 scores for the `MT Epitope Seq` and `WT`
`cterm_7mer_gravy_score`	Mean hydropathy of last 7 residues on the C-terr
`max_7mer_gravy_score`	Max GRAVY score of any kmer in the amino ac
`difficult_n_terminal_residue` (T/F)	Is N-terminal amino acid a Glutamine, Glutamic
`c_terminal_cysteine` (T/F)	Is the C-terminal amino acid a Cysteine?
`c_terminal_proline` (T/F)	Is the C-terminal amino acid a Proline?
`cysteine_count`	Number of Cysteines in the amino acid sequence
`n_terminal_asparagine` (T/F)	Is the N-terminal amino acid a Asparagine?
`asparagine_proline_bond_count`	Number of Asparagine-Proline bonds. Problema
`Best Cleavage Position` (optional)	Position of the highest predicted cleavage score
`Best Cleavage Score` (optional)	Highest predicted cleavage score
`Cleavage Sites` (optional)	List of all cleavage positions and their cleavage s
`Predicted Stability` (optional)	Stability of the pMHC-I complex
`Half Life` (optional)	Half-life of the pMHC-I complex
`Stability Rank` (optional)	The % rank stability of the pMHC-I complex
`NetMHCstab allele` (optional)	Nearest neighbor to the `HLA Allele`. Used fo

3.4.2 filtered.condensed.ranked.tsv Report Columns

Column Name	Description
`Gene Name`	The Ensembl gene names of the affected genes
`Mutation`	NA
`Protein Position`	The position of the fusion in the fusion protein sequence
`HGVSc`	NA
`HGVSp`	NA
`HLA Allele`	The HLA allele for this prediction.
`Mutation Position`	NA
`MT Epitope Seq`	Mutant epitope sequence.
`Median MT Score`	Median ic50 binding affinity of the mutant epitope across all prediction algorithms used
`Median WT Score`	NA
`Median Fold Change`	NA
`Best MT Score`	Lowest ic50 binding affinity of all prediction algorithms used
`Corresponding WT Score`	NA
`Corresponding Fold Change`	NA
`Tumor DNA Depth`	NA
`Tumor DNA VAF`	NA
`Tumor RNA Depth`	NA
`Tumor RNA VAF`	NA
`Gene Expression`	NA
`Rank`	A priority rank for the neoepitope (best = 1).

The pVACfuse Neoeptiope Priority Rank

The underlying formula for calculating the pVACfuse rank is the same as it is for *The pVACseq Neoeptiope Priority Rank*. However, since only the binding affinity is available for fusion predictions, the pVACfuse simply ranks the neoeptiopes according to their binding affinity, with the lowest being the best. If the `--top-score-metric` is set to `median` (default) the `Median MT Score` is used. If it is set to `lowest` the `Best MT Score` is used.

3.5 Filtering Commands

pVACfuse currently offers two filters: a binding filter and a top score filter.

The binding filter and top score filter are always run automatically as part of the pVACfuse pipeline.

All filters can also be run manually to narrow the final results down further or to redefine the filters entirely and produce a new candidate list from the all_epitopes.tsv file.

Note: The default values for filtering thresholds are suggestions only. While they are based on review of the literature and consultation with our clinical and immunology colleagues, your specific use case will determine the appropriate values.

3.5.1 Binding Filter

```
usage: pvacfuse binding_filter [-h] [-b BINDING_THRESHOLD]
                               [-m {lowest,median}] [--exclude-NAs] [-a]
                               input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter.
  output_file           Output .tsv file containing list of filtered epitopes
                        based on binding affinity.

optional arguments:
  -h, --help            show this help message and exit
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the Best MT Score and corresponding Fold Change
                        (i.e. use the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the Median MT Score and Median
                        Fold Change (i.e. use the median MT ic50 binding score
                        and fold change of all chosen prediction methods).
                        (default: median)
  --exclude-NAs         Exclude NA values from the filtered output. (default:
```

(continues on next page)

(continued from previous page)

```
                        False)
  -a, --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
                        allele-specific binding thresholds run `pvacfuse
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
```

The binding filter filters out variants that don't pass the chosen binding threshold. The user can chose whether to apply this filter to the `lowest` or the `median` binding affinity score by setting the `--top-score-metric` flag. The `lowest` binding affinity score is recorded in the `Best MT Score` column and represents the lowest ic50 score of all prediction algorithms that were picked during the previous pVACseq run. The `median` binding affinity score is recorded in the `Median MT Score` column and corresponds to the median ic50 score of all prediction algorithms used to create the report. Be default, the binding filter runs on the `median` binding affinity.

By default, entries with `NA` values will be included in the output. This behavior can be turned off by using the `--exclude-NAs` flag.

3.5.2 Top Score Filter

```
usage: pvacfuse top_score_filter [-h] [-m {lowest,median}]
                                 input_file output_file

positional arguments:
  input_file            The final report .tsv file to filter.
  output_file           Output .tsv file containing only the list of the top
                        epitope per variant.

optional arguments:
  -h, --help            show this help message and exit
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use for filtering. lowest:
                        Use the best MT Score (i.e. the lowest MT ic50 binding
                        score of all chosen prediction methods). median: Use
                        the median MT Score (i.e. the median MT ic50 binding
                        score of all chosen prediction methods). (default:
                        median)
```

This filter picks the top epitope for a variant. Epitopes with the same Chromosome - Start - Stop - Reference - Variant are identified as coming from the same variant.

In order to account for different splice sites among the transcripts of a variant that would lead to different peptides, this filter also takes into account the different transcripts returned by Integrate-Neo/AGFusion and will return the top epitope for all transcripts if they are non-identical. If the resulting list of top epitopes for the transcripts of a variant is identical, the epitope for the transcript with the lowest Ensembl ID is returned.

By default the `--top-score-metric` option is set to `median` which will apply this filter to the `Median MT Score` column and pick the epitope with the lowest median mutant ic50 score for each variant. If the `--top-score-metric` option is set to `lowest`, the `Best MT Score` column is instead used to make this determination.

If there are multiple top epitopes for a variant with the same ic50 score, the first one is chosen.

3.6 Additional Commands

To make using pVACfuse easier, several convenience methods are included in the package.

3.6.1 Download Example Data

```
usage: pvacfuse download_example_data [-h] destination_directory

positional arguments:
  destination_directory
                        Directory for downloading example data

optional arguments:
  -h, --help            show this help message and exit
```

3.6.2 List Valid Alleles

```
usage: pvacfuse valid_alleles [-h]
                              [-p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,
↪NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}]

optional arguments:
  -h, --help            show this help message and exit
  -p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,
↪NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}, --prediction-algorithm {MHCflurry,
↪MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,
↪SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use (default:
                        None)
```

3.6.3 List Allele-Specific Cutoffs

```
usage: pvacfuse allele_specific_cutoffs [-h] [-a ALLELE]

optional arguments:
  -h, --help            show this help message and exit
  -a ALLELE, --allele ALLELE
                        The allele to use (default: None)
```

3.7 Optional Downstream Analysis Tools

3.7.1 Generate Protein Fasta

```
usage: pvacfuse [-h]
                {run,binding_filter,top_score_filter,generate_protein_fasta,valid_
↪alleles,allele_specific_cutoffs,download_example_data}
                ...

positional arguments:
  {run,binding_filter,top_score_filter,generate_protein_fasta,valid_alleles,allele_
↪specific_cutoffs,download_example_data}
    run                 Runs the pVACfuse pipeline
    binding_filter      Filters variants processed by IEDB by binding score
    top_score_filter    Pick the best neoepitope for each variant
    generate_protein_fasta
                        Generate an annotated fasta file from Integrate-Neo or
                        AGFusion output
    valid_alleles       Shows a list of valid allele names
    allele_specific_cutoffs
                        Show the allele specific cutoffs
    download_example_data
                        Downloads example input and output files

optional arguments:
  -h, --help            show this help message and exit
Error: No command specified
```

This tool will extract protein sequences surrounding fusion variant in an by parsing Integrate-Neo or AGFusion output. One use case for this tool is to help select long peptides that contain short neoepitope candidates. For example, if pvacfuse was run to predict nonamers (9-mers) that are good binders and the user wishes to select long peptide (e.g. 24-mer) sequences that contain the nonamer for synthesis or encoding in a DNA vector. The fusion position will be centered in the protein sequence returned (if possible). If the fusion causes a frameshift, the full downstream protein sequence will be returned unless the user specifies otherwise as described above.

pVACvector

CHAPTER 4

pVACvector

pVACvector is designed to aid specifically in the construction of DNA vector based personalized cancer vaccines. It takes as input either a pVACseq output tsv file or a FASTA file containing peptide sequences and returns a peptide ordering that minimizes the effects of junctional epitopes (that may create novel peptides) between the sequences. It does this by using the core pVACseq services to predict the binding scores for each junctional peptide separated by a spacer amino acid sequence that may help to eliminate junctional epitopes. The list of spacers to be tested is specified using the `--spacers` parameter. Peptide combinations without a spacer can be tested by including `None` in the list of spacers.

Peptide junctions are tested with each spacer in the order that they are specified. If a valid peptide ordering is found that doesn't result in any well-binding junction epitopes, that ordering is returned. No other spacer are tested, even if they could potentially result in better junction scores. This reduces runtime. If no valid path is found, the next spacer in the input list is tested. The default spacer amino acid sequences are "None", "AAY", "HHHH", "GGS", "GPGPG", "HHAA", "AAL", "HH", "HHC", "HHH", "HHHD", "HHL", "HHHC".

The final vaccine ordering is achieved through a simulated annealing procedure that returns a near-optimal solution, when one exists.

4.1 Prerequisites

There are two options for the input file when running the pVACvector tool:

- A FASTA file. This file contains protein sequences of candidate neoepitopes to use for vector design.
- A *pVACseq* output TSV. This file has been filtered to include only the neoepitopes to use for vector design. If this file type is used, it is also necessary to provide the original VCF used in the pVACseq run via the `--input_vcf` option. Output TSVs from MHC Class I and Class II pVACseq results can be combined into one by concatenating the two files and removing the duplicate header line.

Note that if you supply a FASTA file of peptides, these peptides will be used directly in the analysis and used in the final output. However, if you use a pvacseq TSV and variants VCF then the length of peptides extracted for junctional epitope testing and reporting in your output will be determined by the `--input-n-mer` option.

4.2 Getting Started

pVACvector provides a set of example data to show the expected input and output files. You can download the data set by running the `pvacfuse download_example_data` *command*.

There are two option as to how to run pVACvector depending on the input file type used. You can either use a pVACseq output TSV of neoepitopes or a FASTA file of peptide sequences.

Here is an example of how to run pVACvector with a pVACseq output TSV:

```
pvacvector run \
<example_data_dir>/input.tsv \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
<output_dir> \
-e 8,9,10 \
-v <example_data_dir>/input.vcf
```

In this example pVACvector is run with an input FASTA file:

```
pvacvector run \
<example_data_dir>/input.fa \
Test \
HLA-A*02:01,HLA-B*35:01,DRB1*11:01 \
MHCflurry MHCnuggetsI MHCnuggetsII NNalign NetMHC PickPocket SMM SMMPMBEC SMMalign \
<output_dir> \
-e 8,9,10
```

A detailed description of all command options can be found on the *Usage* page.

4.3 Usage

Warning: Using a local IEDB installation is strongly recommended for larger datasets or when the making predictions for many alleles, epitope lengths, or prediction algorithms. More information on how to install IEDB locally can be found on the *Installation* page.

It may be necessary to explore the parameter space a bit when running pVACvector. As binding predictions for some sites vary substantially across algorithms, the most conservative settings may result in no valid paths, often due to one "outlier" prediction. Carefully choosing which predictors to run may help ameliorate this issue as well.

In general, setting a lower binding threshold (e.g., 500nM) and using the median binding value (`--top-score-metric median`) will lead to greater possibility of a design, while more conservative settings of 1000nM and lowest/best binding value (`--top-score-metric lowest`) will give more confidence that there are no junctional neoepitopes.

Our current recommendation is to run pVACvector several different ways, and choose the path resulting from the most conservative set of parameters.

```
usage: pvacvector run [-h] [-e EPITOPE_LENGTH]
                      [--iedb-install-directory IEDB_INSTALL_DIRECTORY]
                      [-b BINDING_THRESHOLD]
                      [--allele-specific-binding-thresholds]
                      [-m {lowest,median}] [-r IEDB_RETRIES] [-k]
                      [-t N_THREADS] [-v INPUT_VCF] [-n INPUT_N_MER]
                      [--spacers SPACERS] [--max-clip-length MAX_CLIP_LENGTH]
                      input_file sample_name allele
                      {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}
                      [{MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,
↪NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign} ...]
                      output_dir

positional arguments:
  input_file            A .fa file with peptides or a pVACseq .tsv file with
                        epitopes to use for vector design.
  sample_name           The name of the sample being processed. This will be
                        used as a prefix for output files.
  allele                Name of the allele to use for epitope prediction.
                        Multiple alleles can be specified using a comma-
                        separated list. For a list of available alleles, use:
                        `pvacseq valid_alleles`.
  {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,
↪PickPocket,SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use. Multiple
                        prediction algorithms can be specified, separated by
                        spaces.
  output_dir            The directory for writing all result files.

optional arguments:
  -h, --help            show this help message and exit
  -e EPITOPE_LENGTH, --epitope-length EPITOPE_LENGTH
                        Length of subpeptides (neoepitopes) to predict.
                        Multiple epitope lengths can be specified using a
                        comma-separated list. Typical epitope lengths vary
                        between 8-11. Required for Class I prediction
                        algorithms. (default: None)
  --iedb-install-directory IEDB_INSTALL_DIRECTORY
                        Directory that contains the local installation of IEDB
                        MHC I and/or MHC II. (default: None)
  -b BINDING_THRESHOLD, --binding-threshold BINDING_THRESHOLD
                        Report only epitopes where the mutant allele has ic50
                        binding scores below this value. (default: 500)
  --allele-specific-binding-thresholds
                        Use allele-specific binding thresholds. To print the
```

(continues on next page)

(continued from previous page)

```
                        allele-specific binding thresholds run `pvacvector
                        allele_specific_cutoffs`. If an allele does not have a
                        special threshold value, the `--binding-threshold`
                        value will be used. (default: False)
  -m {lowest,median}, --top-score-metric {lowest,median}
                        The ic50 scoring metric to use when filtering epitopes
                        by binding-threshold or minimum fold change. lowest:
                        Use the best MT Score and Corresponding Fold Change
                        (i.e. the lowest MT ic50 binding score and
                        corresponding fold change of all chosen prediction
                        methods). median: Use the median MT Score and Median
                        Fold Change (i.e. the median MT ic50 binding score and
                        fold change of all chosen prediction methods).
                        (default: median)
  -r IEDB_RETRIES, --iedb-retries IEDB_RETRIES
                        Number of retries when making requests to the IEDB
                        RESTful web interface. Must be less than or equal to
                        100. (default: 5)
  -k, --keep-tmp-files  Keep intermediate output files. This might be useful
                        for debugging purposes. (default: False)
  -t N_THREADS, --n-threads N_THREADS
                        Number of threads to use for parallelizing peptide-MHC
                        binding prediction calls. (default: 1)
  -v INPUT_VCF, --input_vcf INPUT_VCF
                        Path to original pVACseq input VCF file. Required if
                        input file is a pVACseq TSV. (default: None)
  -n INPUT_N_MER, --input-n-mer INPUT_N_MER
                        Length of the peptide sequence to use when creating
                        the FASTA from the pVACseq TSV. (default: 25)
  --spacers SPACERS     Comma-separated list of spacers to use for testing
                        junction epitopes. Include None to test junctions
                        without spacers. Peptide combinations will be tested
                        with each spacer in the order specified. (default: Non
                        e,AAY,HHHH,GGS,GPGPG,HHAA,AAL,HH,HHC,HHH,HHHD,HHL,HHHC
                        )
  --max-clip-length MAX_CLIP_LENGTH
                        Number of amino acids to permit clipping from the
                        start and/or end of peptides in order to test novel
                        junction epitopes when the first pass on the full
                        peptide fails. (default: 3)
```

4.4 Additional Commands

To make using pVACvector easier, several convenience methods are included in the package.

4.4.1 Creating Vector Visualization

By default, pVACvector will create a visualization of the vector design result. For this to happen, the DISPLAY environment variable has to be set. This is often not the case, for example, when running pVACvector on a compute cluster. We provide this convenience method to create the visualization graphic from a successful pVACvector result FASTA file on any machine that has the DISPLAY environment variable set.

```
usage: pvacvector visualize [-h] [-s SPACERS] input_fasta output_directory

positional arguments:
  input_fasta           A pVACvector result FASTA file to visualize
  output_directory      The output directory to save the visualization graphic
                        to

optional arguments:
  -h, --help            show this help message and exit
  -s SPACERS, --spacers SPACERS
                        Comma-separated list of peptides that are used as
                        spacers in the pVACvector result fasta file (default:
                        AAY,HHHH,GGS,GPGPG,HHAA,AAL,HH,HHC,HHH,HHHD,HHL,HHHC)
```

4.4.2 Download Example Data

```
usage: pvacvector download_example_data [-h] destination_directory

positional arguments:
  destination_directory
                        Directory for downloading example data

optional arguments:
  -h, --help            show this help message and exit
```

4.4.3 List Valid Alleles

```
usage: pvacvector valid_alleles [-h]
                                [-p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,
↪NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}]

optional arguments:
  -h, --help            show this help message and exit
  -p {MHCflurry,MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,
↪NetMHCpan,PickPocket,SMM,SMMPMBEC,SMMalign}, --prediction-algorithm {MHCflurry,
↪MHCnuggetsI,MHCnuggetsII,NNalign,NetMHC,NetMHCIIpan,NetMHCcons,NetMHCpan,PickPocket,
↪SMM,SMMPMBEC,SMMalign}
                        The epitope prediction algorithms to use (default:
                        None)
```

4.4.4 List Allele-Specific Cutoffs

```
usage: pvacvector allele_specific_cutoffs [-h] [-a ALLELE]

optional arguments:
  -h, --help            show this help message and exit
  -a ALLELE, --allele ALLELE
                        The allele to use (default: None)
```

4.5 Output Files

File Name	Description
vector_input.fa (optional)	An intermediate file with vaccine peptide sequences created from the epitopes in a pVACseq output file.
<sample_name>_results.fa	The final output file with the peptide sequences and best spacers in the optimal order.
vector.jpg	A JPEG visualization of the above result.

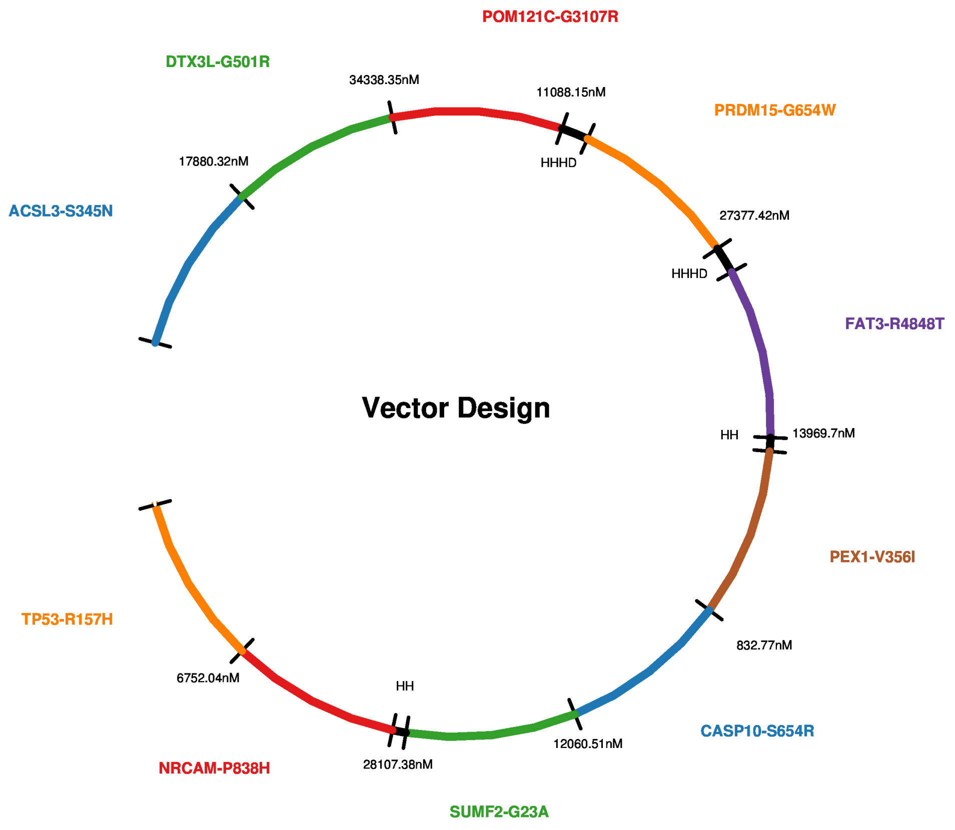


Fig. 1: pVACvector result visualization example

CHAPTER 5

pVACviz

pVACviz is a browser-based, graphical user interface for the pVACtools command line tools. It currently supports starting and managing *pVACseq* runs as well as visualizing the results of your runs.

5.1 Installation

pVACviz is part of the pVACtools package. To install pVACtools, execute the following command on your Terminal:

```
pip install pvactools
```

More detailed installation instructions can be found *here*. Note that the following are the bare minimum you need to run pVACviz. Most users will probably just want to complete the full pvactools install as described *here*. That includes pVACviz along with all the other components, local installation of IEDB, etc. You can also use the pvactools docker container which contains all tools and their dependencies (including those for pVACviz).

5.1.1 MHCflurry

When installing pVACtools for the first time, you will need to manually download the MHCflurry dataset:

```
mhcflurry-downloads fetch
```

5.1.2 PostgreSQL

pVACviz requires a Postgres database. To install Postgres follow the installation instructions.

Note: On Debian-based Linux distributions version Postgres V9.6 or lower is required.

5.2 Running pVACviz

To run pVACviz you first need to start pVACapi, which is used to communicate between the user interface and the command line tool. pVACapi can be started by executing the following command on the command line:

```
pvacapi
```

Depending on the number of completed and running processes it must check, the API may take several seconds to start up. After pVACapi has started, launch pVACviz in a separate terminal window by executing the following command on the command line:

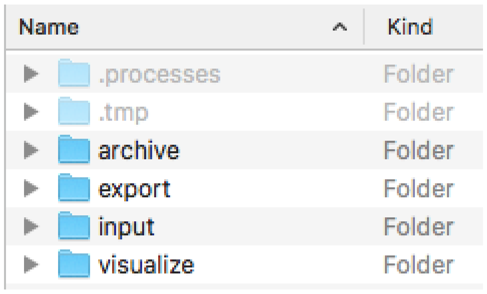
```
pvacviz
```

This command will start a HTTP server that provides the web client files and assets, and opens up the client in the default web browser specified by your operating system. In some cases, pvacviz will not be able to automatically open the web browser. If no browser launches after starting pvacviz, you will need to manually load the URL, *http://localhost:4200/*, in a Firefox, Chrome, or Safari browser. If you are running pVACviz on a public instance, to access it over the web you will need to replace *localhost* with the correct IP address (or associated domain name).

5.3 pVACapi Directories

pVACviz, in order to provide all its features, communicates with a pVACtools component called pVACapi. pVACapi serves as an interface between pVACviz and the pVACseq pipeline proceses, launching pVACseq processes, managing them as they execute, and generating the visualizations that pVACviz displays.

Upon installation, pVACapi creates several directories in the user's home directory in *~/pVAC-Seq/*. These directories are used to hold input files to pVACseq processes, results files for visualization, archives, and exported projects. Additionally two hidden directories controlled by pVACapi are used to store files related to managing and running processes.



Name	Kind
.processes	Folder
.tmp	Folder
archive	Folder
export	Folder
input	Folder
visualize	Folder

Fig. 1: pVACapi directories

5.3.1 /archive

pVACviz provides an archive function within its *Manage section*. When processes are archived they are placed in this archive folder.

5.3.2 /export

pVACviz provides an export function within its *Manage section*. When processes are exported they are placed in this export folder.

5.3.3 /input

The pVACviz *Start form* has Input VCF and Phased Proximal Variant fields that accept VCF files. The selectors for these fields list all relevant files placed within the `~/input` directory. You may sort these files into directories of any depth and the selectors will keep them grouped by directory.

5.3.4 /visualize

The *Visualize feature* allows users to visualize any pVACseq result TSV files. Any pVACseq TSV file placed in this /visualize folder will be displayed on the Visualize page in the right column. Directory structures will be preserved so that users may group files in whatever manner they wish.

5.3.5 /.processes

This is a hidden directory used to store all files related to processes that pVACapi is actively managing. These are the processes listed on the Manage page. You shouldn't touch anything in this directory. Instead, to gain access to these files use the Export or Archive function available in the pVACviz *Manage section*.

5.3.6 /.tmp

This hidden directory is used by pVACapi to store temporary artifacts of the pVACseq pipeline. Editing or deleting anything in this directory may disrupt running pVACseq processes.

5.4 Starting Processes

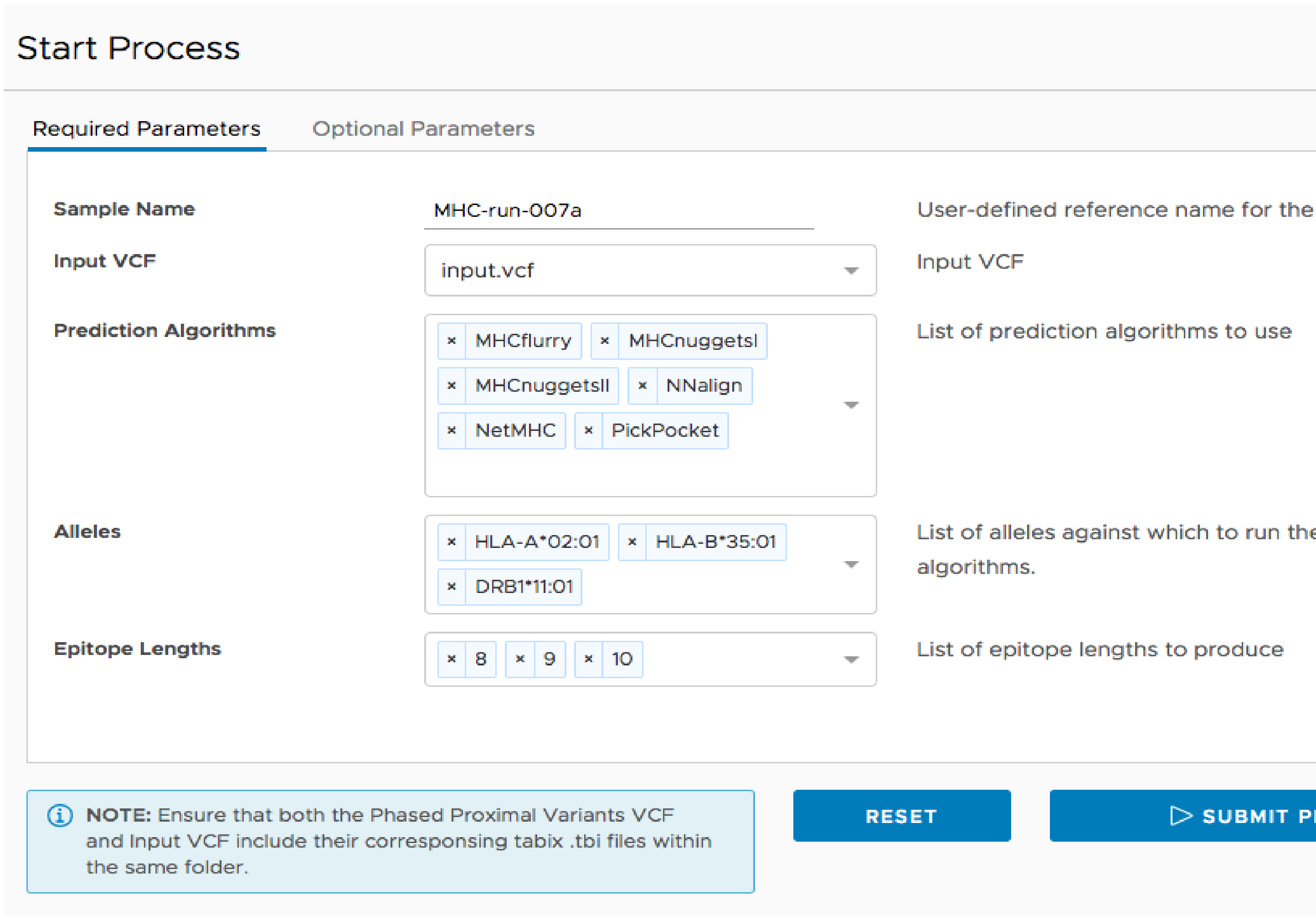
pVACviz provides a helpful form for specifying all of the parameters for a pVACseq process, as an alternative to constructing these commands and executing them via the command line.

5.4.1 Populating and Submitting the Start Form

The form is divided into two sections: required and optional parameters. To be submitted, all required fields must be filled and validated. Optional parameters are pre-filled with sensible defaults - the same defaults that would be applied when submitting processes via the command line.

The form provides feedback as to which fields remain to be filled and validated, both with a red highlight and message around fields in question, and at the bottom of the form with a list of incomplete or invalid fields.

Once all the required form fields are completed with valid values, the Submit Process button is activated. Clicking on this button will submit the process to pVACapi. Clicking the Reset button will restore the form to its initial pristine state.



Sample Name	MHC-run-007a	User-defined reference name for the
Input VCF	input.vcf	Input VCF
Prediction Algorithms	MHCflurry, MHCnuggetsI, MHCnuggetsII, NNalign, NetMHC, PickPocket	List of prediction algorithms to use
Alleles	HLA-A*02:01, HLA-B*35:01, DRB1*11:01	List of alleles against which to run the algorithms.
Epitope Lengths	8, 9, 10	List of epitope lengths to produce

NOTE: Ensure that both the Phased Proximal Variants VCF and Input VCF include their corresponsing tabix .tbi files within the same folder.

Fig. 2: pVACviz start form

5.4.2 Notes

- The Input VCF and Phased Proximal Variant VCF fields require the selection of VCF files. These selectors list all VCF files within the */input folder* found within the ~/pVACseq directory located in the user's home directory.
- The alleles selector only shows alleles relevant to selected prediction algorithms. Choose prediction algorithms to enable and populate the alleles selector.

5.5 Managing Processes

With its management interface users may manage processes launched with pVACviz. The Management section of the application is comprised of two pages: a list page that shows all the currently managed processes; and a detail page that displays all of the details of an individual process.

5.5.1 Displaying All Managed Processes

When you click on the Manage link in the sidebar you will be presented with a table containing a paged list of all currently managed processes.

The process table displays each Process' ID, Status, Sample Name, and Input File and provides a link to the detailed view of each process. A paging interface allows you to page through all running processes.

Each row provides an actions popup menu allowing you to stop, restart, export, archive, and delete processes. Clicking on the three dots at the left side of every row will display a menu of actions that can be applied to the process on that row. See below for a table detailing the actions available to you.

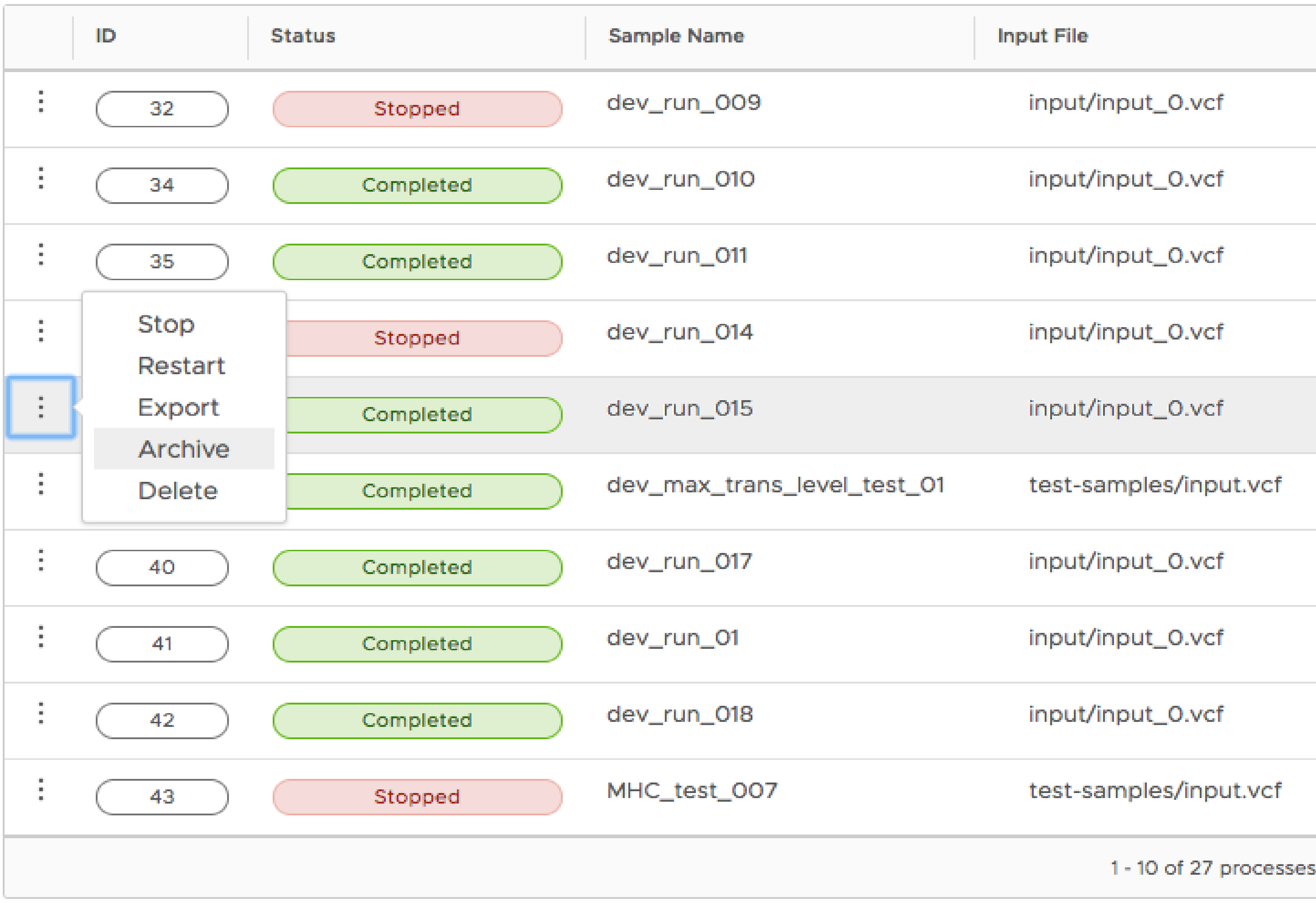
5.5.2 Displaying Process Details

Clicking on the Details link from the main Manage page in a process row displays the Process Detail page shown below. On this page you may view all the details of a process: its log, pVACseq command line arguments, and associated files. Any visualizable files will be shown with a Visualize link which, when clicked, will load the visualization for that results file.

All commands available in the process table are also available here in the header: stop, restart, export, archive and delete. See the table below for more details on these actions.

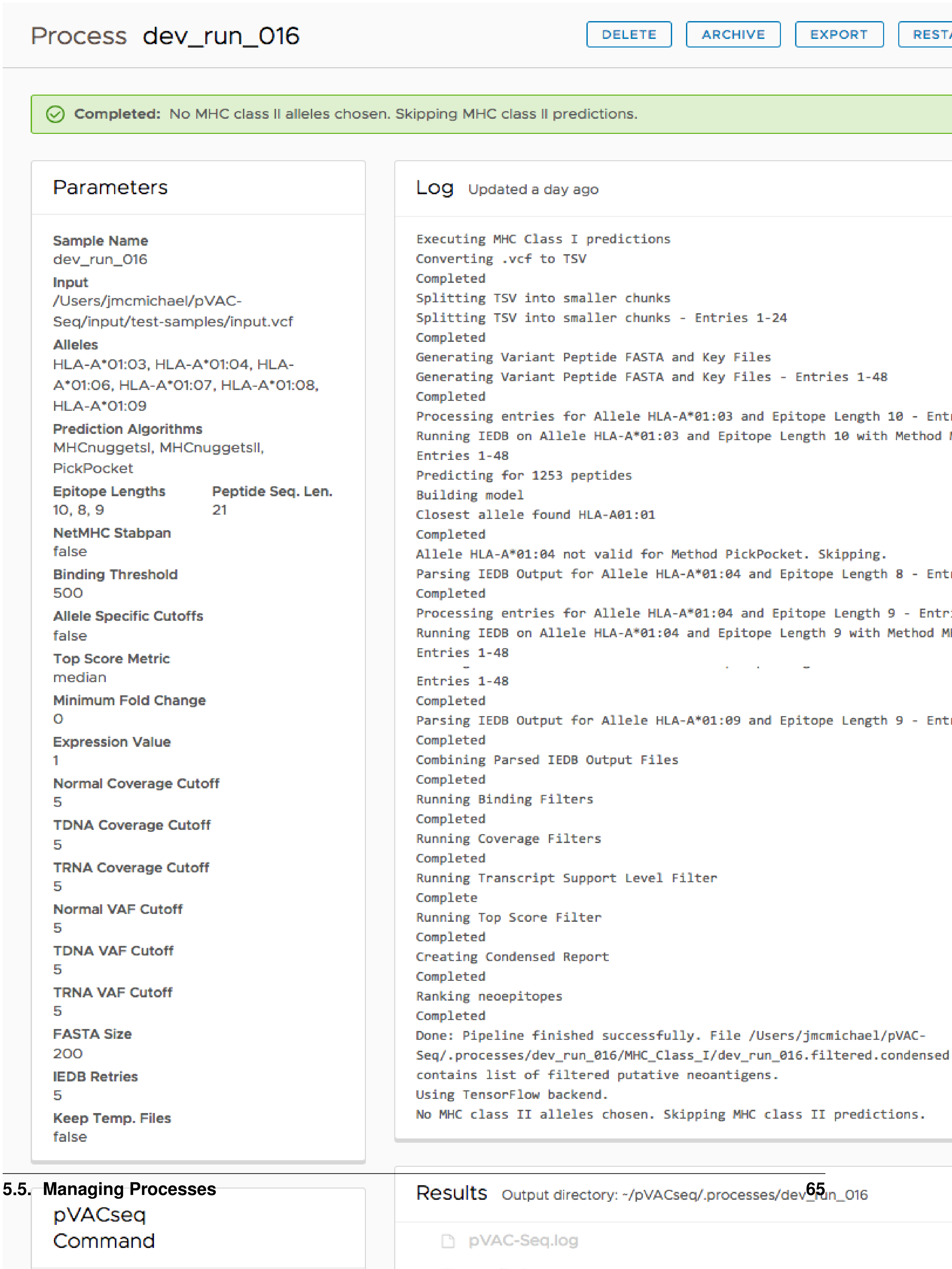
5.5.3 Process Actions

Both the process list table and process detail page provide actions for users to manage pVACseq processes. The process table makes these actions available in its action menu, displayed by clicking on the three dots on the left of every row. The process detail page provides buttons in its header to invoke process actions.



	ID	Status	Sample Name	Input File
⋮	32	Stopped	dev_run_009	input/input_0.vcf
⋮	34	Completed	dev_run_010	input/input_0.vcf
⋮	35	Completed	dev_run_011	input/input_0.vcf
⋮		Stopped	dev_run_014	input/input_0.vcf
⋮		Completed	dev_run_015	input/input_0.vcf
⋮		Completed	dev_max_trans_level_test_01	test-samples/input.vcf
⋮	40	Completed	dev_run_017	input/input_0.vcf
⋮	41	Completed	dev_run_01	input/input_0.vcf
⋮	42	Completed	dev_run_018	input/input_0.vcf
⋮	43	Stopped	MHC_test_007	test-samples/input.vcf

Fig. 3: pVACviz Process List on Manage page



Completed: No MHC class II alleles chosen. Skipping MHC class II predictions.

Parameters

Sample Name
dev_run_016

Input
/Users/jmcmichael/pVAC-Seq/input/test-samples/input.vcf

Alleles
HLA-A*01:03, HLA-A*01:04, HLA-A*01:06, HLA-A*01:07, HLA-A*01:08, HLA-A*01:09

Prediction Algorithms
MHCnuggetsI, MHCnuggetsII, PickPocket

Epitope Lengths
10, 8, 9

Peptide Seq. Len.
21

NetMHC Stabpan
false

Binding Threshold
500

Allele Specific Cutoffs
false

Top Score Metric
median

Minimum Fold Change
0

Expression Value
1

Normal Coverage Cutoff
5

TDNA Coverage Cutoff
5

TRNA Coverage Cutoff
5

Normal VAF Cutoff
5

TDNA VAF Cutoff
5

TRNA VAF Cutoff
5

FASTA Size
200

IEDB Retries
5

Keep Temp. Files
false

Log Updated a day ago

```
Executing MHC Class I predictions
Converting .vcf to TSV
Completed
Splitting TSV into smaller chunks
Splitting TSV into smaller chunks - Entries 1-24
Completed
Generating Variant Peptide FASTA and Key Files
Generating Variant Peptide FASTA and Key Files - Entries 1-48
Completed
Processing entries for Allele HLA-A*01:03 and Epitope Length 10 - Entr
Running IEDB on Allele HLA-A*01:03 and Epitope Length 10 with Method M
Entries 1-48
Predicting for 1253 peptides
Building model
Closest allele found HLA-A01:01
Completed
Allele HLA-A*01:04 not valid for Method PickPocket. Skipping.
Parsing IEDB Output for Allele HLA-A*01:04 and Epitope Length 8 - Entr
Completed
Processing entries for Allele HLA-A*01:04 and Epitope Length 9 - Entr
Running IEDB on Allele HLA-A*01:04 and Epitope Length 9 with Method MH
Entries 1-48
Entries 1-48
Completed
Parsing IEDB Output for Allele HLA-A*01:09 and Epitope Length 9 - Entr
Completed
Combining Parsed IEDB Output Files
Completed
Running Binding Filters
Completed
Running Coverage Filters
Completed
Running Transcript Support Level Filter
Complete
Running Top Score Filter
Completed
Creating Condensed Report
Completed
Ranking neoepitopes
Completed
Done: Pipeline finished successfully. File /Users/jmcmichael/pVAC-
Seq/.processes/dev_run_016/MHC_Class_I/dev_run_016.filtered.condensed
contains list of filtered putative neoantigens.
Using TensorFlow backend.
No MHC class II alleles chosen. Skipping MHC class II predictions.
```

pVACseq Command

Results Output directory: ~/pVACseq/.processes/dev_run_016

pVAC-Seq.log

Action	Description
Stop	Stops a running process. Note that a process must be stopped before it can be restarted, exported, archived, or deleted.
Restart	Restarts a running process. Note that all progress will be lost; pVACseq does not yet restart processes at the point they were stopped.
Export	Exports all a process' config, log, intermediate, and final results files (if any) to the */export directory*. The process will remain in the set of pVACapi managed processes.
Archive	Similar to Export, Archive moves all process config, log, intermediate, and final results files (if any) to the */archive directory*. Unlike Export, Archive removes the process from pVACapi's set of managed processes.
Delete	Deletes all process files and directories and removes it from pVACapi's set of managed proceses. Be careful! This action is not undoable.

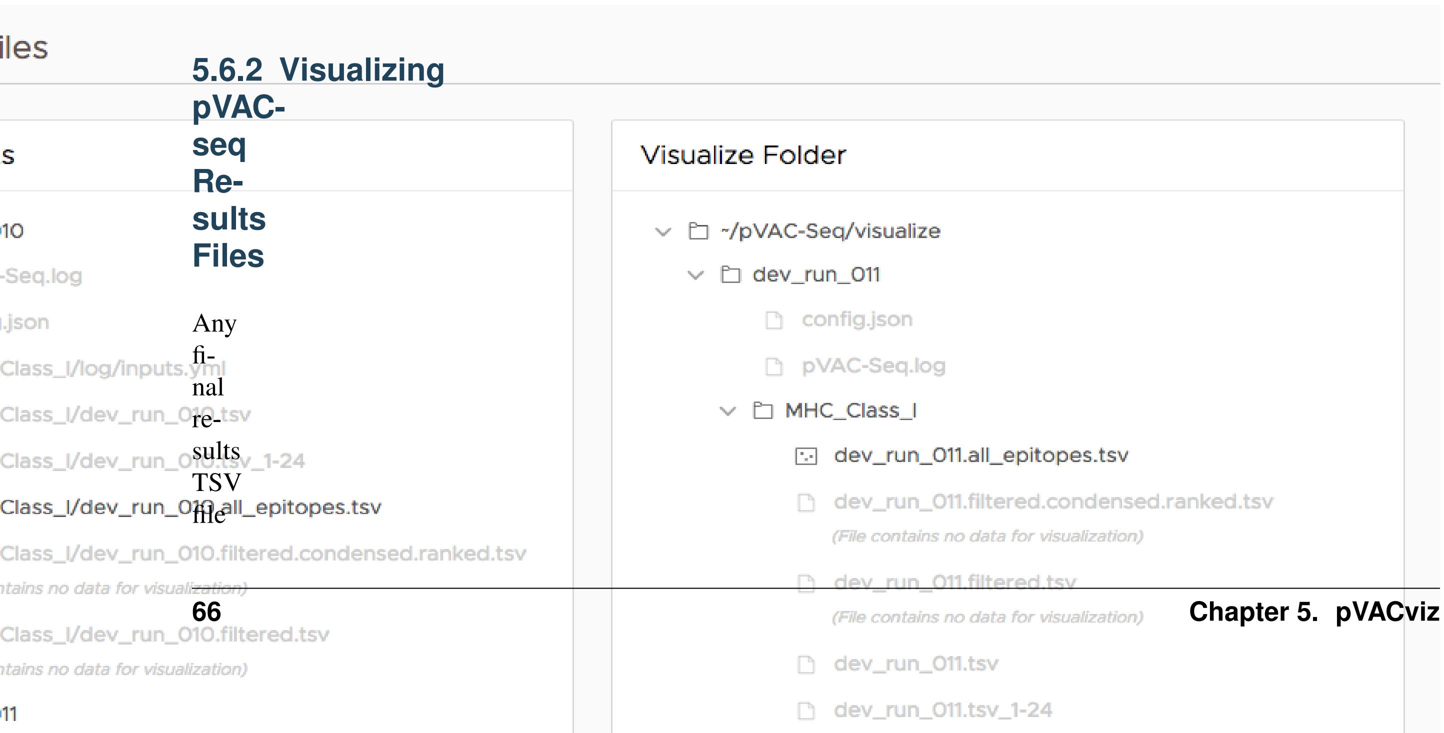
5.6 Visualizing Processes

pVACviz provides a results visualization for exploring the results of pVACseq processes. It is able to visualize both the results from processes launched from pVACviz and results from any pVACseq process.

5.6.1 Visualizing Completed Processes

You may view visualizations of completed pVACseq processes launched from the pVACviz Start form from two locations within the application. The *Manage section* includes a process detail page, reachable by clicking on the Details link on the right side of rows in the process table. On the process detail page, the bottom right card contains a list of all files produced by the pVACseq process. Visualizable files will display a Visualize button that when clicked will load the visualization for that file.

Additionally, on the Visualize main page in the right hand column, all processes currently managed by pVACapi will be listed with their visualizable files. Clicking on a file will load the visualization for that file.



iles

s

10

-Seq.log

.json

Class_I/log/inputs.yml

Class_I/dev_run_010.tsv

Class_I/dev_run_010.tsv_1-24

Class_I/dev_run_010.all_epitopes.tsv

Class_I/dev_run_010.filtered.condensed.ranked.tsv

ntains no data for visualization)

Class_I/dev_run_010.filtered.tsv

ntains no data for visualization)

11

Visualize Folder

~/pVAC-Seq/visualize

dev_run_011

config.json

pVAC-Seq.log

MHC_Class_I

dev_run_011.all_epitopes.tsv

dev_run_011.filtered.condensed.ranked.tsv

(File contains no data for visualization)

dev_run_011.filtered.tsv

(File contains no data for visualization)

dev_run_011.tsv

dev_run_011.tsv_1-24

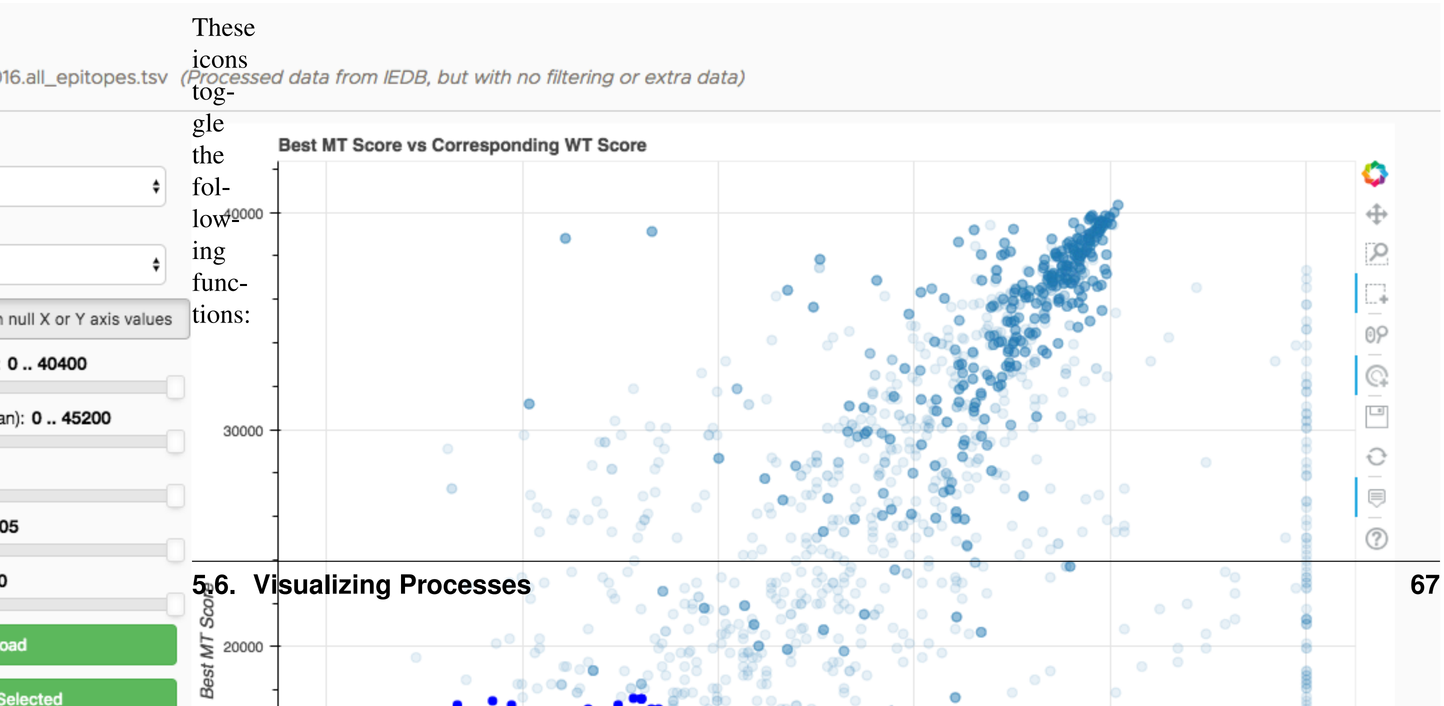
5.6.2 Visualizing pVACseq Results Files

Any final results TSV file

produced by pVACseq processes - whether launched via pVACviz or the command line - may be visualized with pVACviz. You may drop any file or folder in pVACapi's /visualize directory, and it will scan it for visualizable files. These files will then be listed on in the right column of the main Vizualize page. Click on any of the listed pages to launch the visualization.

5.6.3 Scatterplot Visualization

On the visualization's scatterplot are placed all of the data points contained in the tsv results file, one dot per row. A set of tools along the side of the visualization allow you to select and manipulate the plot in various ways. These icons toggle the following functions:


16.all_epitopes.tsv (Processed data from IEDB, but with no filtering or extra data)
Best MT Score vs Corresponding WT Score
null X or Y axis values
0 .. 40400
an): 0 .. 45200
05
0
oad
Selected
40000
30000
20000
Best MT Score

Icon	Name	Function
✥	**Pan**	The pan tool allows the user to pan the plot by left-dragging a mouse or dragging a finger across the plot region.
🔍	**Box Zoom**	The box zoom tool allows the user to define a rectangular region to zoom the plot bounds too, by left-dragging a mouse, or

5.6.4 Axis Columns

To the left of the scatter-plot display are a set of controls that allow you to set the X and Y axis for the scatter-plot and filter the plot's source data. The top two selectors allow you to choose any column of the result set to serve as the X/Y axis.

5.6.5 Filters

You may filter the source data using filters positioned beneath the axis column selectors. By default, points with null X or Y values are filtered out; you may toggle this filter by clicking the Show button. Beneath the show button are displayed a set of sliders that allow you to set min/max values for columns in the result set. Note that the visualization will not show sliders for columns that contain no data.

5.6.6 Data Table

Beneath the plot and filters you will find a datatable linked to both the filters and scatterplot points. Selecting any points in the plot will highlight the corresponsing rows in the data tables. The filters also affect the data table rows - any rows excluded by the filters will also be excluded from the datatable.

5.6.7 Exporting Visualization Data

Two buttons are provided for CSV export of the plot data. The 'Download' button will provide you with a CSV file that contains all of the data provdided to the plot - including filtered rows and rows with null X/Y coordinates. The 'Download Selected' button provides you with a CSV containing only the filtered/selected rows from the plot and datatable.

5.7 pVACapi Troubleshotting

It is possible for pVACapi to get into a state where the cached data contains conflicting information to the actual process outputs. This can be resolved by calling the `pvacapi clear_cache` function on the command line.

CHAPTER 6

Installation

pVACtools is written for Linux but some users have been able to run it successfully on Mac OS X. If you are using Windows you will need to set up a Linux environment, for example by setting up a virtual machine.

pVACtools requires Python 3.5., 3.6, or 3.7. Before running any installation steps, check the Python version installed on your system:

```
python -V
```

If you don't have Python 3.7 installed, we recommend using Conda to emulate a Python 3.7 environment. We've encountered problems with users that already have Python 2.x installed when they also try to install Python 3.7. The defaults will not be set correctly in that case. If you already have Python 2.x installed we **strongly** recommmend using Conda instead of installing Python 3.7 locally.

Once you have set up your Python 3.7 environment correctly you can use `pip` to install pVACtools. Make sure you have `pip` installed. `pip` is generally included in python distributions, but may need to be upgraded before use. See the instructions for installing or upgrading `pip`.

After you have pip installed, type the following command on your Terminal:

```
pip install pvactools
```

You can check that `pvactools` has been installed under the default environment by listing all installed packages:

```
pip list
```

`pip` will fetch and install pVACtools and its dependencies for you. After installing, each tool of the pVACtools suite is available in its own command line tree directly from the Terminal.

If you have an old version of pVACtools installed you might want to consider upgrading to the latest version:

```
pip install pvactools --upgrade
```

6.1 Installing IEDB binding prediction tools (strongly recommended)

Warning: Using a local IEDB installation is strongly recommended for larger datasets or when the making predictions for many alleles, epitope lengths, or prediction algorithms.

Warning: The IEDB binding prediction tools are only compatible with Linux.

You may create a local install of the IEDB binding prediction tools by first downloading the archives for class I and class II from the IEDB website. If using both the Class I and the Class II tools, they both need to be installed into the same parent directory. Note that we have tested pVACtools with the versions of IEDB class I and II listed below. Using a different version may cause problems.

Important: By using the IEDB software, you are consenting to be bound by and become a "Licensee" for the use of IEDB tools and are consenting to the terms and conditions of the Non-Profit Open Software License ("Non-Profit OSL") version 3.0.

Please read these two license agreements here before proceeding. If you do not agree to all of the terms of these two agreements, you must not install or use the product. Companies (for-profit entities) interested in downloading the command-line versions of the IEDB tools or running the entire analysis resource locally, should contact IEDB (license@iedb.org) for details on licensing options.

Citing the IEDB

All publications or presentations of data generated by use of the IEDB Resource Analysis tools should include citations to the relevant reference(s), found here.

Note: Using a local IEDB install with pVACtools requires conda.

pVACtools is written in python 3 and IEDB is only compatible with python 2.7. Because of this version mismatch, the pVACtools modules will create a custom python 2.7 environment and execute IEDB inside of it. This requires conda.

6.1.1 MHC Class I

Download the archives for class I and unpack them.

```
apt-get update && apt-get install -y tcsh gawk
wget https://downloads.iedb.org/tools/mhci/2.19.2/IEDB_MHC_I-2.19.2.tar.gz
tar -zxvf IEDB_MHC_I-2.19.2.tar.gz
cd mhc_i
./configure
```

Note: Running the `configure` script requires a Python 2 environment. If you are currently emulating a Python 3 environment with Conda you will need to run `source deactivate` before executing the `configure` script.

6.1.2 MHC Class II

Download the archives for class II and unpack them.

```
apt-get update && apt-get install -y tcsh gawk
wget https://downloads.iedb.org/tools/mhcii/2.17.6/IEDB_MHC_II-2.17.6.tar.gz
tar -zxvf IEDB_MHC_II-2.17.6.tar.gz
cd mhc_ii
./configure.py
```

On older versions of the IEDB software, you might need to update some paths in the configure scripts to use relative paths. Open the `configure.py` file and update the lines that set the `smm` and `nn` variables to use relative paths like so:

```
smm = re.compile(curDir + "/netMHCII-1.1")
nn = re.compile(curDir + "/netMHCII-2.2")
```

Then run the configure script.

```
./configure.py
```

Note: Running the `configure` script requires a Python 2 environment. If you are currently emulating a Python 3 environment with Conda you will need to run `source deactivate` before executing the `configure` script.

6.2 Installing MHCflurry

If you wish to run the MHCflurry prediction algorithm, you will need to install the `mhcflurry` python package on your system. This package is set as a dependency for the `pvactools` package so it should be installed automatically when you download or upgrade the `pvactools` package. You can install it manually by running:

```
pip install mhcflurry
```

Note: The `mhcflurry` package needs to be installed in the same python 3 conda environment as the `pvactools` package.

Next, you will need to download the download the MHCflurry datasets and trained models:

```
mhcflurry-downloads fetch
```

Note: The `mhcflurry-downloads fetch` command will need to be run manually, even if the mhcflurry package was already installed automatically as a dependency with the `pvactools` package.

You can check that the `mhcflurry` package was installed successfully by running:

```
mhcflurry-predict -h
```

This should pull up the help page for the MHCflurry predictor.

Please note that MHCflurry depends on tensorflow, which will automatically be installed as a dependency to the `mhcflurry` package. Newer versions of tensorflow might not be compatible with older CPUs. In that case you will see a core dump failure. Downgrading tensorflow manually to version 1.5.0 should solve this problem:

```
pip install tensorflow==1.5.0
```

6.3 Installing MHCnuggets

If you wish to run the MHCnuggets prediction algorithm, you will need to install the `mhcnuggets` python package on your system. This package is set as a dependency for the `pvactools` package so it should be installed automatically when you download or upgrade the `pvactools` package. You can install it manually by running:

```
pip install mhcnuggets
```

Note: The `mhcnuggets` package needs to be installed in the same python 3 conda environment as the `pvactools` package.

You can check that the `mhcnuggets` package was installed successfully by running:

```
pip show mhcnuggets
```

This should show information about the mhcnuggets package.

Please note that MHCnuggets depends on tensorflow, which will automatically be installed as a dependency to the `mhcnuggets` package. Newer versions of tensorflow might not be compatible with older CPUs. In that case you will see a core dump failure. Downgrading tensorflow manually to version 1.5.0 should solve this problem:

```
pip install tensorflow==1.5.0
```

6.4 PostgreSQL

pVACviz and pVACapi require a Postgres database. To install Postgres follow the installation instructions.

Note: On Debian-based Linux distributions version Postgres V9.6 or lower is required.

6.5 Docker and CWL

A Docker container for pVACtools is available on DockerHub using the griffithlab/pvactools repo. This Docker container includes installations of the IEDB class I and class II tools at `/opt/iedb` (`--iedb-install-directory /opt/iedb`).

An example on how to run pVACseq using Docker can be found on the *Getting Started* page.

Common Workflow Language (CWL) tool wrappers for pVACseq, pVACfuse, and pVACvector can be downloaded using the `pvactools download_cwls` command.

6.5.1 Download CWL tool wrappers

```
usage: pvactools download_cwls [-h] destination_directory

positional arguments:
  destination_directory
                        Directory for downloading CWLs

optional arguments:
  -h, --help            show this help message and exit
```

CHAPTER 7

Tools Used By pVACtools

7.1 IEDB (Immune Epitope Database)

- Website: https://www.iedb.org
- Citation: Vita R, Mahajan S, Overton JA, Dhanda SK, Martini S, Cantrell JR, Wheeler DK, Sette A, Peters B. The Immune Epitope Database (IEDB): 2018 update. Nucleic Acids Res. 2018 Oct 24. doi: 10.1093/nar/gky1006. [Epub ahead of print] PubMed PMID: 30357391.
- License: Non-Profit OSL 3.0

  By using the IEDB software, you are consenting to be bound by and become a "Licensee" for the use of IEDB tools and are consenting to the terms and conditions of the Non-Profit Open Software License ("Non-Profit OSL") version 3.0.

  Please read these two license agreements here before proceeding. If you do not agree to all of the terms of these two agreements, you must not install or use the product. Companies (for-profit entities) interested in downloading the command-line versions of the IEDB tools or running the entire analysis resource locally, should contact IEDB (license@iedb.org) for details on licensing options.

7.2 MHCflurry

- Website: http://openvax.github.io/mhcflurry/
- GitHub: https://github.com/openvax/mhcflurry
- Citation: T. J. O'Donnell, A. Rubinsteyn, M. Bonsack, A. B. Riemer, U. Laserson, and J. Hammerbacher, "MHCflurry: Open-Source Class I MHC Binding Affinity Prediction," Cell Systems, 2018. doi: https://doi.org/10.1016/j.cels.2018.05.014. PubMed PMID: 29960884.
- License: Apache License 2.0

7.3 MHCnuggets

- Website: https://karchinlab.org/apps/appMHCnuggets.html
- GitHub: https://github.com/KarchinLab/mhcnuggets-2.0
- Citation: Bhattacharya R, Sivakumar A, Tokheim C, Beleva Guthrie V, Anagnostou V, Velculescu VE, Karchin R (2017) Evaluation of machine learning methods to predict peptide binding to MHC Class I proteins. Submitted [bioRxiv preprint].
- License: Apache License 2.0

7.4 NetChop

- Website: http://www.cbs.dtu.dk/services/NetChop/
- Citation: The role of the proteasome in generating cytotoxic T cell epitopes: Insights obtained from improved predictions of proteasomal cleavage. M. Nielsen, C. Lundegaard, O. Lund, and C. Kesmir. Immunogenetics., 57(1-2):33-41, 2005. PubMed PMID: 11983929.
- License: Academic License

7.5 NetMHCstabpan

- Website: http://www.cbs.dtu.dk/services/NetMHCstabpan/
- Citation: Pan-specific prediction of peptide-MHC-I complex stability; a correlate of T cell immunogenicity. Michael Rasmussen, Emilio Fenoy, Mikkel Harndahl, Anne Bregnballe Kristensen, Ida Kallehauge Nielsen, Morten Nielsen, Soren Buus. J Immunol. 2016 Aug 15;197(4):1517-24. doi: https://doi.org/10.4049/jimmunol.1600582. PubMed PMID: 27402703.
- License: Academic License

7.6 Vaxrank

- Website: https://github.com/openvax/vaxrank
- Citation: Rubinsteyn, A., Hodes, I., Kodysh, J., & Hammerbacher, J. (2017). Vaxrank: a computational tool for designing personalized cancer vaccines. bioRxiv, 142919.
- License: Apache License 2.0

CHAPTER 8

Frequently Asked Questions

How is pVACtools licensed?

pVACtools is licensed under NPOSL-3.0.

Where can I get help?

Bug reports or feature requests can be submitted on the pVACtools Github page. You may also contact us by email at help@pvactools.org.

How do I cite pVACtools?

Jasreet Hundal+, Susanna Kiwala+, Joshua McMichael, Christopher A Miller, Alexander T Wollam, Huiming Xia, Connor J Liu, Sidi Zhao, Yang-Yang Feng, Aaron P Graubert, Amber Z Wollam, Jonas Neichin, Megan Neveau, Jason Walker, William E Gillanders, Elaine R Mardis, Obi L Griffith, Malachi Griffith. pVACtools: a computational toolkit to select and visualize cancer neoantigens. bioRxiv 501817; doi: https://doi.org/10.1101/501817. (+)equal contribution.

Jasreet Hundal, Susanna Kiwala, Yang-Yang Feng, Connor J. Liu, Ramaswamy Govindan, William C. Chapman, Ravindra Uppaluri, S. Joshua Swamidass, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. Accounting for proximal variants improves neoantigen prediction. Nature Genetics. 2018, DOI: 10.1038/s41588-018-0283-9. PMID: 30510237.

Jasreet Hundal, Beatriz M. Carreno, Allegra A. Petti, Gerald P. Linette, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. pVACseq: A genome-guided in silico approach to identifying tumor neoantigens. Genome Medicine. 2016, 8:11, DOI: 10.1186/s13073-016-0264-5. PMID: 26825632.

CHAPTER 9

Release Notes

9.1 Version 1.0

9.1.1 1.0.0

This is the initial release of pVACtools, a cancer immunotherapy suite consisting of the following tools:

pVACseq

A cancer immunotherapy pipeline for identifying and prioritizing neoantigens from a list of tumor mutations.

pVACfuse

A tool for detecting neoantigens resulting from gene fusions.

pVACvector

A tool designed to aid specifically in the construction of DNA vector-based cancer vaccines.

9.1.2 1.0.1

This is a hotfix release. It fixes the following issues:

- Additional data, like example data and VEP plugins were not included in the package correctly so the commands to download these files would fail. This has been corrected.
- Class II predictions would fail if the protein sequences used for binding predictions in IEDB were shorter than 15 peptide sequences. This has been fixed.

9.1.3 1.0.2

This is a hotfix release. It fixes the following issues:

- The epitope length used for generating the peptide fasta when running with multiple epitope lengths was incorrect. This would potentially result in including fasta sequences that were shorter than the largest epitope length which would cause an error during calls to IEDB.
- pVACseq would fail with a nondescript error message if the input VCF was not annotated with VEP before running. A more descriptive error message has been added.
- IEDB changed the format of class II IEDB alleles which would cause an error when running with those alleles. pVACtools will now handle transposing the affected alleles into the new format.
- The standalone binding filters had a few bugs that would result in syntax errors during runtime.
- The indexes created for each fusion entry with pVACfuse had the potential to not be unique which would result in parsing errors downstream.
- pVACseq had the potential to use the incorrect VEP allele for positions with multiple alternate alleles which would result in the incorrect CSQ entry getting used for some of those alternate alleles.
- pVACseq would throw an error if the chosen peptide sequence length exceeds the wildtype protein sequence length of a transcript.

9.1.4 1.0.3

This is a hotfix release. It fixes the following issues:

- Stop-gain mutation were previously not handled correctly. If a mutation had a * (stop gain) in the VEP Amino_acids field, pVACseq would throw an error. We now ensure that those cases are handled. pVACseq will also skip stop-gain mutations if the resulting mutant peptide sequence is not novel.
- pVACseq would previously throw an error if multiple mutations resulted in the same consequence. This is now handled by assigning a unique identifier to each mutation.
- We added a better warning messages if the chosen prediction algorithms and alleles MHC classes are mutually exclusive, e.g., if only class I prediction algorithms were chosen with only class II alleles. Previously, pVACseq would simply finish without producing any output or errors.

9.1.5 1.0.4

This is a hotfix release. It fixes the following issues:

- We discovered a couple more cases of mutations involving stop codons that would result in errors. These are amino acid changes (VEP Amino_acids field) for large indels that would span exon boundaries (multiple * in the Amino_acids field), or amino acid changes involving the transcript stop codon (ending in X). These cases are now handled.

9.1.6 1.0.5

This is a hotfix release. It fixes the following issues:

- IEDB changed the format of combinatorial class II alleles to use `/` as a delimiter instead of `-`. DP alleles were previously fixed in pull request #85 but this failed to address DQ alleles. This version fixes this oversight.

9.1.7 1.0.6

This is a hotfix release. It fixes the following issues:

- There was a bug in how alternate alleles were resolved when matching VEP consequence fields to an entry which resulted in certain indels to be skipped. This has now been fixed.

9.1.8 1.0.7

This is a hotfix release. It fixes the following issues:

- VEP82 and higher supports selenocysteine modicfications (amino acid "U"), which is not supported by downstream IEDB prediction algorithms. pVACtools now skips sequences containing this amino acid with a warning.

9.1.9 1.0.8

This is a hotfix release. It fixes the following issues:

- The log directories were accidentially included with the pVACseq example data. They are now removed.
- Some users were reporting mixed type warnings for pandas when running pVACseq. We added some options to avoid this warning.

9.2 Version 1.1

9.2.1 1.1.0

This version adds a host of new features to pVACtools:

- pVACseq is now able to parse VAF, depth, and expression information directly from the VCF. This makes the `--additional-input-file-list` option obsolete. The `--additional-input-file-list` option is now deprecated and will be removed in an upcoming release. For more information on how to annotate your VCF with readcount and expression information, see the *Input File Preparation* page.
- pVACseq is now able to handle proximal germline and somatic variants. In order to incorporate those into the epitope predictions, you will need to provide a phased variants VCF to your pVACseq run using the `--phased-proximal-variants-vcf` option. For more information on how to create this file, see the *Input File Preparation* page.
- We added support to pVACseq for filtering on transcript support levels. This requires the input VCF to be annotated with the TSL field by VEP. Be default, any transcripts with a TSL above 1 will be filtered out.
- The binding filter of pVACseq and pVACfuse can now be run with flexible, allele-specific binding-thresholds. This feature can be enabled using the `--allele-specific-binding-thresholds` flag. The thresholds used are taken from the IEDB recommendations.
- pVACseq now supports a `--pass-only` flag that will result in any VCF entries with a `FILTER` to be skipped. Using this flag, only VCF entries with a `FILTER` of `PASS` or `.` will be processed.
- We added support for the MHCflurry and MHCnuggets prediction algorithms. These can be used by listing `MHCflurry`, `MHCnuggetsI` (for MHC Class I alleles), and/or `MHCnuggetsII` (for MHC Class II alleles) as the prediction algorithms in your run commands.
- The default `--tdna-vaf` and `--trna-vaf` cutoff values have been updated from 0.4 to 0.25. This is the minimum VAF threshold that an epitope candidate must meet in order to pass the coverage filter.
- We now offer a graphical user interface, *pVACviz*, to run pVACseq as an alernative to using the command line. pVACviz, can also be used to plot and filter your pVACseq results.

9.2.2 1.1.1

This is a hotfix release. It fixes the following issue(s):

- In version 1.1 we updated VAFs to be fractions, rather than percentages. A bug in this code change resulted in an error when using custom VAF cutoff values instead of the default. This has now been fixed.

9.2.3 1.1.2

This is a hotfix release. It fixes the following issue(s):

- In version 1.1.0 we added a `--pass-only` flag to pVACseq that would result in only variants with `FILTER` of `PASS` or `.` getting processed. However, this option was not getting passed along to the pVACseq process correctly, resulting in this option not taking effect. This hotfix release fixes this issue and the `--pass-only` flag should now work as expected.

9.2.4 1.1.3

This is a hotfix release. It fixes the following issue(s):

- When using the MHCnuggets prediction algorithm for MHC class II alleles (`MHCnuggetsII`) not all epitope sequences were predicted for inframe insertions. This issues has now been fixed.
- For MHCflurry, cases with peptide sequences that were shorter than the desired epitope length were not handled correctly which resulted in an error. This issue has been resolved in this release.

9.2.5 1.1.4

This is a hotfix release. It fixes the following issue(s):

- When running pVACvector with a with a pVACseq input file and the corresponding VCF, the sample name wasn't being passed along correctly which would cause an error if the input VCF was a multi-sample VCF.
- pVACseq would throw an error if the value of a gene or transcript expression field was empty.

9.2.6 1.1.5

This is a hotfix release. It fixes the following issue(s):

- When running pVACseq with a phased input VCF the mutation position offset of a frameshift somatic variant to their proximal variants was not getting calculated correctly, leading to errors.
- For running pVACvector we removed a dependency on a commandline tool by using a python library instead. This allowed us to remove a system call to a tool that required standalone installation by the user.

9.3 Version 1.2

9.3.1 1.2.0

This version introduces multiprocessing to pVACtools. This significantly speeds up the execution of pVACseq, pVAC-fuse, and pVACvector. To turn on multiprocessing simply set the `--n-threads` parameter to the desired number of parallel processes. When running the tools using the IEDB RESTful API, we recommend to keep this number small

(<5) as too many parallel calls to their API might lead to IEDB blocking jobs submitted from your IP address. It is recommended to use a standalone IEDB installation when running in multiprocessing mode. By default, multiprocessing is turned off.

This version also fixes a few bugs:

- In certain cases pVACvector was not calculating the junction scores correctly, leading to potentially finding a peptide order that would include high-binding junction epitopes or peptide orders that were not optimal. This issue has now been fixed.
- Due to a bug in our packaging code, the 1.1.x versions of pVACtools did not include the latest version of the pVACviz code. This version now includes the most up-to-date version of the graphical user interface.

9.4 Version 1.3

9.4.1 1.3.0

This version adds a few features and updates:

- pVACvector now accepts a list of spacers to use when testing junction epitopes. These can be specified using the `--spacers` parameter with a comma-separated list of spacer peptides. Including the string `None` will also test each junction without spacers. The default is `None,HH,HHC,HHH,HHHD,HHHC,AAY,HHHH,HHAA,HHL,AAL`
- The `--expn-val` cutoff parameter has been updated to be a float instead of an integer. This allows the user to provide a decimal cutoff for the filtering on gene and transcript expression values. Previously, only whole numbers were accepted.
- Decimal numbers in the pVACseq reports are now rounded to three decimal places. Previously, they were not rounded.

In addition, this version also fixes a few bugs:

- The `--normal-vaf` cutoff value was incorrectly defaulting to 0.2 instead of 0.02. This resulted in the coverage filter not being as stringent as it should've been.
- There were a number of bugs in pVACapi and pVACviz that would prevent a user from submitting jobs using the interface in certain conditions. These have been resolved.
- pVACseq would previously not support SVs in the input VCF where the alt had a value of `<DEL>`. These kinds of variants are now supported.

9.4.2 1.3.1

This version is a hotfix release. It fixes the following issues:

- Some prediction algorithms might predict a binding affinity of 0 which could lead to division by 0 errors when calculating the fold change. In this situation we now set the fold change to `inf` (infinity).
- Previously the `--maximum-transcript-support-level` threshold was not getting propagated to the main pipeline step correctly, resulting in errors in the transcript support level filter.
- There was a bug in the multiprocessing logic that would result in certain steps getting executed more than once, which in turn would lead to FileNotFound errors when these duplicate executions were happening at the same time.

9.4.3 1.3.2

This version is a hotfix release. It fixes the following issues:

- A bug in the parsing code of the binding prediction output files would result in only some binding prediction output files getting processed when using multiprocessing. This would potentially cause incomplete output reports that were missing predictions for some input variants. pVACseq, pVACfuse, and pVACvector runs that were done without multiprocessing should've been unaffected by this bug.

9.4.4 1.3.3

This version is a hotfix release. It fixes the following issues:

- We were previously using our own locking logic while running in multiprocssing mode which contained a bug that could result in runs getting stuck waiting on a lock. This release switches to using the locking implementation provided by the `pymp-pypi` multiprocessing package.
- In an attempt to reduce cluttered output generated by Tenserflow we were previously repressing any message generated during the import of MHCflurry and MHCnuggets. As a side effect, this would also suppress any legitimate error messages generated during these imports which would result in the `pvacseq`, `pvacfuse`, and `pvacvector` commands exiting without output. This release updates to code so that actual errors still get output.

9.4.5 1.3.4

This version is a hotfix release. It fixes the following issues:

- We were previously using nested multiprocessing which would cause defunct child jobs and stalled runs. Switching to single-level multiprocessing fixes this issue.
- When running pVACvector from a pVACseq result file the creation of the peptide fasta file might cause an error if the epitope was situated near the beginning of the transcript. This issue has been fixed.

9.4.6 1.3.5

This version is a hotfix release. It fixes the following issues:

- While the previous release fixed the issue of stalled processes when running IEDB-based prediction algorithms in multiprocessing mode, we were still experience a similar problem when running with MHCflurry and MHCnuggets. These two prediction algorithms are tensorflow-based which in the way it is currently used in pVACtools is not compatible with being run in multiprocessing mode. As a stop-gap measure this release removes MHCnuggets and MHCflurry from being run in multiprocessing mode. This resolves the problem until we can change our usage of these prediction algorithms to be multiprocessing-compatible.

9.4.7 1.3.6

This version is a hotfix release. It fixes the following issues:

- Tensorflow is incompatible with multiprocessing when the parent process imports tensorflow or a tensorflow-dependent module. For this reason MHCflurry and MHCnuggets were removed from parallelization. In this release we moved to calling MHCflurry and MHCnuggets on the command line, which allowed us to remove our direct imports of these modules and allows us to parallelize the calls to these two prediction algorithms. All prediction algorithms supported by pVACtools can now be used in multiprocessing mode.

- Some users were reporting `Illegal instruction (core dumped)` errors because their hardware was incompatible with the version of tensorflow we were using. Pinning the tensorflow version to 1.5.0 with this release should solve this problem.
- When running in multiprocessing mode while using the IEDB API, users would experience a higher probability of failed requests to the API. The IEDB API would throw a 403 error when rejecting requests due to too many simultaneous requests. pVACtools would previously not retry on this type of error. This release now adds retries on this error code. We also improved the random wait time calculation between requests so that the likelihood of multiple retries hitting at the same time has now been reduced.
- When encountering a truncated input VCF, the VCF parser used by pVACtools would throw an error that was not indicative of the real error source. pVACseq now catches these errors and emmits a more descriptive error message when encountering a truncated VCF.
- One option when annotating a VCF with VEP is the `-total-length` flag. When using this flag, the total length would be written to the `Protein_position` field. pVACseq previously did not support a VCF with a `Protein_position` field in this format. This release adds support for it.
- When creating the combined MHC class I and MHC class II all_epitopes file, we were previously not correctly determining all necessary headers which would lead to incorrect output of the individual prediction algorithm score columns. This release fixes this issue.

9.4.8 1.3.7

This version is a hotfix release. It fixes the following issues:

- The previous version accidentally removed the `--additional-input-file-list` option. It has been restored in this version. Please note that it is slated for permanent removal in the next feature release (1.4.0).

9.5 Version 1.4

9.5.1 1.4.0

This version adds the following features:

- pVACvector now tests spacers iteratively. During the first iteration, the first spacer in the list of `--spacers` gets tested. In the next iteration, the next spacer in the list gets added to the pool of spacers to tests, and so on. If at any point a valid ordering is found, pVACvector will finish its run and output the result. This might result in a slightly less optimal (but still valid) ordering but improves runtime significantly.
- If, after testing all spacers, no valid ordering if found, pVACvector will clip the beginning and/or ends of problematic peptides by one amino acid. The ordering finding process is then repeated on the updated list of peptides. This process may be repeated a number of times, depending on the value of the `--max-clip-length` parameter.
- This version adds a standalone command to create the pVACvector visualizations that can be run by calling `pvacvector visualize` using a pVACvector result file as the input.
- We removed the `--aditional-input-file-list` option to pVACseq. Readcount and expression information are now taken directly from the VCF annotations. Instructions on how to add these annotations to your input VCF can be found on the *Input File Preparation* page.
- We added support for variants to pVACseq that are only annotated as `protein_altering_variant` without a more specific consequence of `missense_variant`, `inframe_insertion`, `inframe_deletion`, or `frameshift_variant`.

- We resolved some syntax differences that prevented pVACtools from being run under python 3.6 or python 3.7. pVACtools should now be compatible with all python >= 3.5 versions.

9.5.2 1.4.1

This is a hotfix release. It fixes the following issues:

- In version 1.4 we updated our usage of conda to use `conda activate` instead of `source activate` to make it compatible with newer conda versions. However, this was leading to errors due to the way that we were calling conda. This has been updated and should resolve these types of errors.

9.5.3 1.4.2

This is a hotfix release. It fixes the following issues:

- This releases fixes a concurrency issue with pVACapi/pVACviz that would occurr when users would try to visualize multiple files at the same time.

9.5.4 1.4.3

This is a hotfix release. It fixes the following issues:

- IEDB will output a warning if an epitope contains only amino acid symbols that could also be nucleotides. This would cause an error during parsing of the IEDB output files. This version updates the parser to ignore these warnings.
- We added some improvements to pVACapi regarding database file read speeds and transaction handling.

9.5.5 1.4.4

This is a hotfix release. It fixes the following issues:

- This version starts enforcing a file size limit (14MB) to be able to visualize a result file in pVACviz. Larger files will no longer be visualizable in pVACviz since they take too long to load.

9.5.6 1.4.5

This is a hotfix release. It fixes the following issues:

- In a previous version we implemented a faster method for reading data from the database in pVACapi. However, this would fail if the postgres user is not a superuser. This version fixes this issue by using the previous database file read method in this situation.
- This version marks certain columns of the output reports as not visualizable in pVACviz/pVACapi because they contain string content that cannot be plotted in a scatterplot.

9.6 Version 1.5

9.6.1 1.5.0

This version adds the following features:

- This version introduces a new tool, `pVACbind`, which can be used to run our immunotherapy pipeline with a peptides FASTA file as input. This new tool is similar to pVACseq but certain options and filters are removed:
  - All input sequences are interpreted in isolation so corresponding wildtype sequence and score information are not assigned. As a consequence, the filter threshold option on fold change is removed.
  - Because the input format doesn't allow for association of readcount, expression or transcript support level data, pVACbind doesn't run the coverage filter or transcript support level filter.
  - No condensed report is generated.

  Please see the *pVACbind* documentation for more information.
- pVACfuse now support annotated fusion files from AGFusion as input. The *pVACfuse* documentation has been updated with instructions on how to run AGFusion in the Prerequisites section.
- The top score filter has been updated to take into account alternative known transcripts that might result in non-indentical peptide sequences/epitopes. The top score filter now picks the best epitope for every available transcript of a variant. If the resulting list of epitopes for one variant is not identical, the filter will output all eptiopes. If the resulting list of epitopes for one variant are identical, the filter only outputs the epitope for the transcript with the highest transcript expression value. If no expression data is available, or if multiple transcripts remain, the filter outputs the epitope for the transcripts with the lowest transcript Ensembl ID.
- This version adds a few new options to the `pvacseq generate_protein_fasta` command:
  - The `--mutant-only` option can be used to only output mutant peptide sequences instead of mutant and wildtype sequences.
  - This command now has an option to provide a pVACseq all_eptiopes or filtered TSV file as an input (`--input-tsv`). This will limit the output fasta to only sequences that originated from the variants in that file.
- This release adds a `pvacfuse generate_protein_fasta` command that works similarly to the `pvacseq generate_protein_fasta` command but works with Integrate-NEO or AGFusion input files.
- We removed the sorting of the all_epitopes result file in order to reduce memory usage. Only the filtered files will be sorted. This version also updates the sorting algorithm of the filtered files as follows:
  - If the `--top-score-metric` is set to `median` the results are first sorted by the `Median MT Score`. If multiple epitopes have the same `Median MT Score` they are then sorted by the `Corresponding Fold Change`. The last sorting criteria is the `Best MT Score`.
  - If the `--top-score-metric` is set to `lowest` the results are first sorted by the `Best MT Score`. If multiple epitopes have the same `Best MT Score` they are then sorted by the `Corresponding Fold Change`. The last sorting criteria is the `Median MT Score`.
- pVACseq, pVACfuse, and pVACbind now calculate manufacturability metrics for the predicted epitopes. Manufacturability metrics are also calculated for all protein sequences when running the `pvacseq generate_protein_fasta` and `pvacfuse generate_protein_fasta` commands. They are saved in the `.manufacturability.tsv` along to the result fasta.
- The pVACseq score that gets calculated for epitopes in the condensed report is now converted into a rank. This will hopefully remove any confusion about whether the previous score could be treated as an absolute measure of immunogencity, which it was not intended for. Converting this score to a rank ensures that it gets treated in isolation for only the epitopes in the condensed file.
- The condensed report now also outputs the mutation position as well as the full set of lowest and median wildtype and mutant scores.
- This version adds a clear cache function to pVACapi that can be called by running `pvacapi clear_cache`. Sometimes pVACapi can get into a state where the cache file contains conflicting data compared to the actual

process outputs which results in errors. Clearing the cache using the `pvacapi clear_cache` function can be used in that situation to resolve these errors.

9.6.2 1.5.1

This is a hotfix release. It fixes the following issues:

- There was a syntax error in `tools/pvacseq/generate_condensed_ranked_report.py` that would result in an error when running the `pvacseq generate-condensed-ranked-report` commands.
- We were previously not correctly catching cases where the intermediate fasta for making binding prediction was empty. This would result in errors downstream.

9.6.3 1.5.2

This is a hotfix release. It fixes the following issues:

- AGFusion exon files may be comma-delimited. Previously, the file parser assumed the files were tab-delimited. This release now allows AGFusion inputs that are comma- or tab-delimited.

9.6.4 1.5.3

This is a hotfix release. It fixes the following issues:

- pVACbind would previously throw an error if a peptide sequence in the input fasta was shorter than one of the chosen epitope lengths. This issue has been fixed by first parsing the input fasta and creating individual fasta files for each epitope length that enforce a minimum length of the peptide sequences matching the respective epitope length.
- Previous versions of pVACtools resolved an issue where IEDB would output a warning line if one of the epitope sequences only contained A, C, G, or T amino acids, since those sequences could also be nuclotide sequences. However, this issue was only fixed in pVACseq, not pVACbind, or pVACvector. This release fixes this issue for all tools.
- The wrappers for NetChop or NetMHCstabpan split the set of input epitopes into chunks of 100 before processing. Due to a bug in the file splitting logic, one epitope for each chunk over 100 would be errenously dropped. This effectively would result in less epitopes being returned in the filtered report than if running the pipelines without NetChop or NetMHCstabpan.

9.6.5 1.5.4

This is a hotfix release. It fixes the following issues:

- The `pvacseq generate_protein_fasta` command would previously error out when running with a selected `peptide_sequence_length` that would reduce in peptides < 7 amino acids long. This error would occur when calculating manufacturability metrics. This release now only calculates these metrics for peptides >=7 amino acids long.
- We updated the calculation for the flanking sequence length when generating peptide sequences to result in peptides that are closer in length to the requested `peptide_sequence_length`.
- This release fixes an edge case where a frameshift mutation impacted the first amino acid of a transcript. This case would previously throw a fatal error but will now be processed as expected.

9.6.6 1.5.5

This is a hotfix release. It fixes the following issues:

- The `pvacfuse run` command would previously output a misleading warning message if an AGFusion input directory didn't contain any processable fusion entries. This warning message has been fixed.
- Between VEP versions, the Downstream protein sequence prediction for some frameshift mutations was changed to now include a leading wildtype amino acid. This potential difference in VEP-predicted Downstream protein sequences was not accounted for and would result in frameshift mutation protein prediction that would duplicate this leading wildtype amino acid. This version updates our prediction pipeline to remove this duplicated amino acid and output a fatal error if the Downstream protein sequence does not contain the leading wildtype amino acid.

9.6.7 1.5.6

This is a hotfix release. It fixes the following issues:

- The `pvacbind run` command would previously error out if one of the input sequences would contain a X stop codon. This update will remove the X amino acid and the downstream sequence before further processing the remaining protein sequence.
- A bug in the `pvacfuse top_score_filter` code would previsouly result in an error when trying to run this command. This has now been fixed.

9.6.8 1.5.7

This is a hotfix release. It fixes the following issues:

- The `pvacbind run` command would previously allow fasta input files with duplicated headers. However, it would silently skip subsequent entries with duplicated headers even if the fasta sequence was novel. With this release pVACbind will now error out if a duplicate fasta header is encounterd.

CHAPTER 10

Citations

Jasreet Hundal+, Susanna Kiwala+, Joshua McMichael, Christopher A Miller, Alexander T Wollam, Huiming Xia, Connor J Liu, Sidi Zhao, Yang-Yang Feng, Aaron P Graubert, Amber Z Wollam, Jonas Neichin, Megan Neveau, Jason Walker, William E Gillanders, Elaine R Mardis, Obi L Griffith, Malachi Griffith. pVACtools: a computational toolkit to select and visualize cancer neoantigens. Cancer Immunology Research. 2020 Mar;8(3):409-420. DOI: 10.1158/2326-6066.CIR-19-0401. PMID: 31907209. (+) equal contribution.

Jasreet Hundal, Susanna Kiwala, Yang-Yang Feng, Connor J. Liu, Ramaswamy Govindan, William C. Chapman, Ravindra Uppaluri, S. Joshua Swamidass, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. Accounting for proximal variants improves neoantigen prediction. Nature Genetics. 2018, DOI: 10.1038/s41588-018-0283-9. PMID: 30510237.

Jasreet Hundal, Beatriz M. Carreno, Allegra A. Petti, Gerald P. Linette, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. pVACseq: A genome-guided in silico approach to identifying tumor neoantigens. Genome Medicine. 2016, 8:11, DOI: 10.1186/s13073-016-0264-5. PMID: 26825632.

CHAPTER 11

Contact

Bug reports or feature requests can be submitted on the pVACtools Github page. You may also contact us by email at help@pvactools.org.

To stay up-to-date on the latest pVACtools releases please join our *Mailing List*.

CHAPTER 12

Mailing List

To stay up-to-date on the latest pVACtools releases please join our mailing list by browsing to https://groups.google.com/forum/#!forum/pvactools-users and clicking the blue "Join group to post" button.

CHAPTER 13

New in release 1.5.7

This is a hotfix release. It fixes the following issues:

- The `pvacbind run` command would previously allow fasta input files with duplicated headers. However, it would silently skip subsequent entries with duplicated headers even if the fasta sequence was novel. With this release pVACbind will now error out if a duplicate fasta header is encounterd.

CHAPTER 14

New in version 1.5

This version adds the following features:

- This version introduces a new tool, `pVACbind`, which can be used to run our immunotherapy pipeline with a peptides FASTA file as input. This new tool is similar to pVACseq but certain options and filters are removed:
  - All input sequences are interpreted in isolation so corresponding wildtype sequence and score information are not assigned. As a consequence, the filter threshold option on fold change is removed.
  - Because the input format doesn't allow for association of readcount, expression or transcript support level data, pVACbind doesn't run the coverage filter or transcript support level filter.
  - No condensed report is generated.

  Please see the *pVACbind* documentation for more information.
- pVACfuse now support annotated fusion files from AGFusion as input. The *pVACfuse* documentation has been updated with instructions on how to run AGFusion in the Prerequisites section.
- The top score filter has been updated to take into account alternative known transcripts that might result in non-indentical peptide sequences/epitopes. The top score filter now picks the best epitope for every available transcript of a variant. If the resulting list of epitopes for one variant is not identical, the filter will output all eptiopes. If the resulting list of epitopes for one variant are identical, the filter only outputs the epitope for the transcript with the highest transcript expression value. If no expression data is available, or if multiple transcripts remain, the filter outputs the epitope for the transcripts with the lowest transcript Ensembl ID.
- This version adds a few new options to the `pvacseq generate_protein_fasta` command:
  - The `--mutant-only` option can be used to only output mutant peptide sequences instead of mutant and wildtype sequences.
  - This command now has an option to provide a pVACseq all_eptiopes or filtered TSV file as an input (`--input-tsv`). This will limit the output fasta to only sequences that originated from the variants in that file.
- This release adds a `pvacfuse generate_protein_fasta` command that works similarly to the `pvacseq generate_protein_fasta` command but works with Integrate-NEO or AGFusion input files.

- We removed the sorting of the all_epitopes result file in order to reduce memory usage. Only the filtered files will be sorted. This version also updates the sorting algorithm of the filtered files as follows:
  - If the `--top-score-metric` is set to `median` the results are first sorted by the `Median MT Score`. If multiple epitopes have the same `Median MT Score` they are then sorted by the `Corresponding Fold Change`. The last sorting criteria is the `Best MT Score`.
  - If the `--top-score-metric` is set to `lowest` the results are first sorted by the `Best MT Score`. If multiple epitopes have the same `Best MT Score` they are then sorted by the `Corresponding Fold Change`. The last sorting criteria is the `Median MT Score`.
- pVACseq, pVACfuse, and pVACbind now calculate manufacturability metrics for the predicted epitopes. Manufacturability metrics are also calculated for all protein sequences when running the `pvacseq generate_protein_fasta` and `pvacfuse generate_protein_fasta` commands. They are saved in the `.manufacturability.tsv` along to the result fasta.
- The pVACseq score that gets calculated for epitopes in the condensed report is now converted into a rank. This will hopefully remove any confusion about whether the previous score could be treated as an absolute measure of immunogencity, which it was not intended for. Converting this score to a rank ensures that it gets treated in isolation for only the epitopes in the condensed file.
- The condensed report now also outputs the mutation position as well as the full set of lowest and median wildtype and mutant scores.
- This version adds a clear cache function to pVACapi that can be called by running `pvacapi clear_cache`. Sometimes pVACapi can get into a state where the cache file contains conflicting data compared to the actual process outputs which results in errors. Clearing the cache using the `pvacapi clear_cache` function can be used in that situation to resolve these errors.

Past release notes can be found on our *Release Notes* page.

To stay up-to-date on the latest pVACtools releases please join our *Mailing List*.

CHAPTER 15

Citations

Jasreet Hundal , Susanna Kiwala , Joshua McMichael, Chris Miller, Huiming Xia, Alex Wollam, Conner Liu, Sidi Zhao, Yang-Yang Feng, Aaron Graubert, Amber Wollam, Jonas Neichin, Megan Neveau, Jason Walker, William Gillanders, Elaine Mardis, Obi Griffith, Malachi Griffith. pVACtools: A Computational Toolkit to Identify and Visualize Cancer Neoantigens. Cancer Immunology Research. 2020 Mar;8(3):409-420. doi: 10.1158/2326-6066.CIR-19-0401. PMID: 31907209.

Jasreet Hundal, Susanna Kiwala, Yang-Yang Feng, Connor J. Liu, Ramaswamy Govindan, William C. Chapman, Ravindra Uppaluri, S. Joshua Swamidass, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. Accounting for proximal variants improves neoantigen prediction. Nature Genetics. 2018, DOI: 10.1038/s41588-018-0283-9. PMID: 30510237.

Jasreet Hundal, Beatriz M. Carreno, Allegra A. Petti, Gerald P. Linette, Obi L. Griffith, Elaine R. Mardis, and Malachi Griffith. pVACseq: A genome-guided in silico approach to identifying tumor neoantigens. Genome Medicine. 2016, 8:11, DOI: 10.1186/s13073-016-0264-5. PMID: 26825632.

CHAPTER 16

Source code

The pVACtools source code is available in GitHub.

CHAPTER 17

License

This project is licensed under NPOSL-3.0.