epitopepredict Documentation

Damien Farrell

Jul 20, 2019
## Contents

1 Introduction 3

2 Installation 5
   2.1 Prediction algorithms 6
   2.2 Installing netMHCpan and netMHCIIpan 6
   2.3 Installing IEDB MHC-I tools 6

3 Submit Bugs 9

4 References 11

5 Command Line Interface 13
   5.1 Usage 13
   5.2 Configuration file settings 13
   5.3 Settings explained 14
   5.4 Cutoff methods 15
   5.5 Binding promiscuity 15
   5.6 Preset allele lists 15
   5.7 IEDB tool methods 15
   5.8 Examples 16
   5.9 Outputs 17

6 Web Application 19
   6.1 Usage and Interface 19
   6.2 Future features 20

7 Code Examples 23
   7.1 Methodology 23
   7.2 Basics 24
   7.3 Analysis 25

8 Neoepitope Prediction 27
   8.1 About 27
   8.2 Method 28
   8.3 Usage and Configuration 28
   8.4 References 29

9 MHC Allele Nomenclature 31
10 File Formats

11 epitopepredict
   11.1 epitopepredict package .......................................................... 35

12 Indices and tables ........................................................................... 59

Python Module Index ........................................................................... 61

Index ...................................................................................................... 63
epitopepredict provides a standardized programmatic and command line interface for executing multiple MHC binding prediction methods. The results from each method can then be processed and visualized in a consistent manner.
This software should be run on a Linux operating system. Ubuntu is recommended but most major distributions will be fine. Windows is not supported. macOS (OS X) may work but has not been tested. If you use windows or a mac you can simply install a linux virtual machine and run from there. You can then run the command line interface or use in python. Install with pip using:

```
sudo pip install epitopepredict
```

**Snap package**

You can install as a snap on linux which might be easier than using pip for some users. This option is convenient because updates can be provided regularly and automatically. Almost everything you need is packaged in the snap. Snaps are ubuntu based but supported on most major linux distributions. Install `snapd` with your package manager and then run:

```
sudo snap install epitopepredict
```

Or you can just go to [https://snapcraft.io/epitopepredict](https://snapcraft.io/epitopepredict) and install from there.

Updating the snap is automatic but can be done manually using:

```
sudo snap refresh epitopepredict
```

*Note: netMHC prediction methods are not packaged in the snap because of restrictions in how those programs are redistributed. This is addressed below in the section on installing netMHCpan.*

**Python dependencies**

- numpy
- pandas
- matplotlib
- biopython
- tornado
- bokeh
2.1 Prediction algorithms

There are now multiple MHC binding prediction algorithms available freely online. Often the problem is determining how to use them and which alleles they support. The ‘state of the art’ algorithms are probably those based on neural networks such as netMHC class I and II routines. These are packaged as external tools and can be installed freely on your system. Some of the algorithms below are included in the snap package so you don’t have to install them separately.

Supported algorithms

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
<th>snap package</th>
</tr>
</thead>
<tbody>
<tr>
<td>tepitope</td>
<td>implements the TEPITOPEPan method, built in (MHC-II)</td>
<td>yes</td>
</tr>
<tr>
<td>netMHCpan</td>
<td><a href="http://www.cbs.dtu.dk/services/NetMHCpan/">http://www.cbs.dtu.dk/services/NetMHCpan/</a> (MHC-I)</td>
<td>not yet</td>
</tr>
<tr>
<td>netMHCIIpan</td>
<td><a href="http://www.cbs.dtu.dk/services/NetMHCIIpan/">http://www.cbs.dtu.dk/services/NetMHCIIpan/</a> (MHC-II)</td>
<td>not yet</td>
</tr>
<tr>
<td>mhcflurry</td>
<td><a href="https://github.com/openvax/mhcflurry">https://github.com/openvax/mhcflurry</a> (MHC-I)</td>
<td>yes</td>
</tr>
<tr>
<td>mhcnuggets</td>
<td><a href="https://github.com/KarchinLab/mhcnuggets-2.0">https://github.com/KarchinLab/mhcnuggets-2.0</a> (MHC-I/II)</td>
<td>yes</td>
</tr>
<tr>
<td>IEDB MHC-I tools</td>
<td><a href="http://tools.immuneepitope.org/mhci/download/">http://tools.immuneepitope.org/mhci/download/</a></td>
<td>no</td>
</tr>
</tbody>
</table>

Both mhcflurry and mhcnuggets can be installed easily with pip and are provided as part of the snap package.

2.2 Installing netMHCpan and netMHCIIpan

Due to license restrictions these programs must be installed separately. You can go to these respective links to fill in forms that will give you access to the install file:

- http://www.cbs.dtu.dk/cgi-bin/nph-sw_request?netMHCpan
- http://www.cbs.dtu.dk/cgi-bin/nph-sw_request?netMHCIIpan

The install instructions can then be found in the readme files when you untar the downloaded file e.g. netMHCpan-4.0.readme. Remember to test the software is working before you use it in epitopepredict.

2.2.1 Using netMHCpan with the snap package

Since netMHCpan can’t be distributed with the snap, you have to download it separately. For snaps, netMHCpan should be installed in a folder called tools in your home directory. That is all you should have to do and the program will be found at run time.

2.3 Installing IEDB MHC-I tools

Note that if using the netMHCpan programs above you probably do not need to use the IEDB tools unless you have specific requirements to do so. The distributions ‘IEDB_MHC*.tar.gz’ contain a collection of peptide binding prediction tools for Major Histocompatibility Complex (MHC) class I and II molecules. The collection is a mixture of pythons scripts and linux 32-bit environment specific binaries. Linux environment is required. Under ubuntu (if not using the snap package) you should also install tcsh and gawk:
sudo apt install tcsh gawk

Download from http://tools.iedb.org/mhci/download/. Unpack the tar.gz files. Run the ‘configure’ script to set up path variables for trained models. This has been tested to work with version 2.17.

tar -zxvf IEDB_MHC_I-*.tar.gz
     cd mhc_i
     ./configure.py

*MHC-II tools are not currently supported.*
Submit Bugs

This software is under active development particularly with a view to improve the command line tools. Please use the github project page to submit bugs or suggestions: http://dmnfarrell.github.io/epitopepredict
CHAPTER 4

References


CHAPTER 5

Command Line Interface

Installing the package provides the command `epitopepredict` in your path. This is a command line interface to the library without the need for any Python coding. It provides pre-defined functionality with settings specified in a text configuration file. Using this you can make MHC predictions with your chosen alleles and predictors. If you are using the IEDB prediction tools they should be installed locally and you can specify the path in the [iedbtools] section. Otherwise ignore those settings. Note that if settings are left out generally defaults will be used so you can have a minimal file as in the examples.

5.1 Usage

Usage largely involves setting up the config file and having your input files prepared. Running the command `epitopepredict -c <yourfilename>.conf` will create a new config file for you to work from if it doesn’t exist. Just edit this with a text editor and then to execute:

```bash
epitopepredict -c <yourfilename>.conf -r
```

You can also test the pipeline after installing by running:

```bash
epitopepredict -t
```

This will generate predictions using a set of sample HIV-1 sequences and save the results to a folder called hiv1_test which you can open in the web app to view (see below). This should work ‘out of the box’ as it only uses the built in prediction algorithm, tepitope.

5.2 Configuration file settings

The advantage of configuration files is in avoiding long commands that have to be remembered or are prone to mistakes. Also the config files can be kept to recall what setting we used or to copy them for another set of files. The current options available in the file are shown below:
5.3 Settings explained

<table>
<thead>
<tr>
<th>name</th>
<th>example value</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>predictors</td>
<td>tepitope</td>
<td>name of predictor: e.g. tepitope, iedbmhc1, netmhciipan, mhcfllurry</td>
</tr>
<tr>
<td>mhc1_alleles</td>
<td>HLA-A<em>01:01,HLA-A</em>03:01</td>
<td>list of MHC-I alleles or preset name</td>
</tr>
<tr>
<td>mhc2_alleles</td>
<td>HLA-DRB1<em>0101,HLA-DRB1</em>0103</td>
<td>list of MHC-II alleles or preset name</td>
</tr>
<tr>
<td>mhc1_length</td>
<td>11</td>
<td>length of n-mers for MHC-I prediction</td>
</tr>
<tr>
<td>mhc2_length</td>
<td>11</td>
<td>length of n-mers for MHC-II prediction</td>
</tr>
<tr>
<td>n</td>
<td>3</td>
<td>minimum number of alleles for counting as promiscuous binders</td>
</tr>
<tr>
<td>cutoff_method</td>
<td>score</td>
<td>cutoff method: default, score or rank used for getting binders          (see below)</td>
</tr>
<tr>
<td>cutoffs</td>
<td>.95</td>
<td>percentile/score/rank cutoff for counting binders</td>
</tr>
<tr>
<td>sequence_file</td>
<td>zaire-ebolavirus.gb</td>
<td>set of protein sequences in genbank or fasta format</td>
</tr>
<tr>
<td>peptide_file</td>
<td>peptides.txt</td>
<td>set of peptides in a plain text file, one per row</td>
</tr>
<tr>
<td>path</td>
<td>results</td>
<td>folder to save results to, can be empty for current folder</td>
</tr>
<tr>
<td>overwrite</td>
<td>no</td>
<td>overwrite the previous results</td>
</tr>
<tr>
<td>names</td>
<td>Rv0011c,Rv0019c</td>
<td>subset of protein/sequence names to predict from your input file, optional</td>
</tr>
<tr>
<td>verbose</td>
<td>no</td>
<td>displays more information while running</td>
</tr>
<tr>
<td>cpus</td>
<td>1</td>
<td>number of processors to use, use 0 for all available</td>
</tr>
<tr>
<td>iedbmhc1_path</td>
<td></td>
<td>folder where the IEDB MHC-I tools are installed, not required unless used</td>
</tr>
<tr>
<td>iedb_mhc1_method</td>
<td>IEDB_recommended</td>
<td>predictor to use within the IEDB MHC-I tools (see below)</td>
</tr>
</tbody>
</table>
5.4 Cutoff methods

Methods for achieving an appropriate cutoff for considering a peptide to be a binder are somewhat arbitrary. They vary with the application. There are three methods provided to select binders:

- **default** - allele specific global cutoffs, this uses a percentile cutoff to select peptides using pre-calculated quantile scores for each allele. This may avoid an issue where certain alleles will dominate if using a single score cutoff. Though there is limited evidence to suggest this is more appropriate. Typical value would be .95 i.e. top 95% in each allele.

- **rank** - Select top ranking peptides in each sequence above the cutoff. This would be useful for small numbers of sequence but for a lot of proteins might produce too many false positives.

- **score** - Use a single score cutoff for all peptides/alleles. This is probably the standard method. Typical binding predictors produce an affinity score and a cutoff of 500 is used. However this might also produce a lot of false positives.

5.5 Binding promiscuity

Promiscuous binders are those above the cutoffs in more than n alleles. The rationale for this is that a peptide is more likely to be immunogenic in your target population if it is a binder in multiple alleles. This may not be the case in reality of course. By default the command line tool will calculate the promiscuous binders to give you a unique list of peptides and include the number of alleles in which it is a binder. The table is ranked by this value and the maximum score over the alleles tested.

5.6 Preset allele lists

For convenience there are some lists of common alleles that you can use without having to type allele names into the config file. These have been taken from various sources and are only a rough guide. Use `epitopepredict -p` to see the available presets. The format of allele names is discussed on the MHC Allele Nomenclature page.

The current selection is:

<table>
<thead>
<tr>
<th>name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mhc1_supertypes</td>
<td>6 MHC-I supertypes</td>
</tr>
<tr>
<td>mhc2_supertypes</td>
<td>7 MHC-II supertypes</td>
</tr>
<tr>
<td>us_caucasion_mhc1</td>
<td>30 most common US caucasion MHC-I</td>
</tr>
<tr>
<td>us_african_mhc1</td>
<td>30 most common US african MHC-I</td>
</tr>
<tr>
<td>human_common_mhc2</td>
<td>11 most prevalent HLA-DR alleles worldwide</td>
</tr>
<tr>
<td>broad_coverage_mhc1</td>
<td>26 alleles providing broad coverage</td>
</tr>
<tr>
<td>bovine_like_mhc2</td>
<td>8 HLA-DR alleles chosen to approximate bovine response</td>
</tr>
</tbody>
</table>

5.7 IEDB tool methods

The IEDB combines multiple prediction methods into its tools. Generally it’s recommended to use their IEDB_recommended method but individual methods may be preferred. You can specify these using the `iedb_mhc1_method` option. Remember they do not all support all alleles. See Installing IEDB MHC-I tools.
5.8 Examples

MHC-II binding predictions for preset alleles of proteins in a genbank file

Using preset allele lists saves you the trouble of writing the alleles out. You can get the built-in presets by using -p at the command line. If you provide MHC-I alleles for a class II predictor like tepitope the program will give an error.

More cpus means speed improvements:

```
[base]
predictors = tepitope
mhc2_alleles = human_common_mhc2
n = 2
cutoffs = .95
sequence_file = zaire-ebolavirus.gb
path = results
cpus = 2
```

A small set of peptides

Say we want to predict for small list of peptides with multiple prediction methods and select the top 10 ranking in at least 3 alleles. Here input.txt is just simple text file with all the individual peptides. They should be of an appropriate length:

```
[base]
predictors = tepitope,mhcflurry
mhc1_alleles = human_common_mhc2
mhc2_alleles = human_common_mhc2
cutoff_method = rank
cutoffs = 10
n=3
path = results
peptide_file = input.txt
```

Strict cutoffs

For selection you can use very strict score cutoff level or high global percentile. In this example we use a score cutoff so must provide a cutoff value for each method:

```
[base]
predictors = tepitope,netmhcipan
mhc1_alleles = human_common_mhc2
cutoff_method = score
cutoffs = 6,50
n=3
path = results
peptide_file = input.txt
```
5.9 Outputs

In each results folder you will find a sub-folder for each method. This has csv files with the predictions for each sequence, if using multiple protein sequences. This is the primary raw output. These folders can be re-used as input in the analysis section without re-running predictions and read by the web interface for presentation if needed. There are also files of the form final_method_n.csv which contain the promiscuous binders for each method.
A web app that is launched from the command line can be used to view and analyze results from a set of predictions that you have made. This is an improved and much easier to use form of a previous web interface called epitopemap and replaces it. Note: this app is still under development, suggestions for additional functionality are welcome. In the future it will probably be possible to launch predictions inside the app. For now you run predictions on the command line and view them in the browser.

6.1 Usage and Interface

After you have made some predictions you can run the app (usually from the same folder where you ran your predictions) using:

```
epitopepredict -s
```

The default port is 8888. You can use a different port by specifying it with the -x option. This should open a new web page in your browser. To view all results from a run you then enter the path (folder) where your results were saved in the form and press submit. This should refresh your form with a drop down list of all the available sequences/proteins.

There are several ways to view a set of binding predictions, all of which allow views for whichever predictors you have used. There is currently

**Summary view**

Summarizes the results for multiple sequences in one page. You can choose to view the table of all predicted binders, promiscuous binders or a summary over each sequence. These tables can be downloaded to csv files.

**Sequence view**

For viewing the detailed results for a single sequence, often representing a protein coding sequence. Graphical views of the prediction scoring across the sequence are designed to provide a quick look at the pattern of peptide binding prediction in multiple alleles. By default a track view of each allele/predictor is shown as below:
Track plots are useful for overall results over protein-length and longer sequences. The plot can be zoomed and panned using the mouse. A hover tooltip shows the particular peptide details.

Grid plots are another way to view peptide scores across a protein sequence for each allele. A hover tooltip shows the particular peptide details.

**Config**

Allows you to generate a configuration file from a form for running a set of predictions. In future this could be used to submit jobs directly.

### 6.2 Future features

- Improved graphical features for genome based prediction.
- Location of clusters of binders in sequences.
- Export of peptide lists/n-mers for experimental use.
- Mutation/conservation analysis.
• Edit config and run predictions from web page.
CHAPTER 7

Code Examples

This page is for those using the Python API. For those wanting to use the command line application see the Command line interface page. General usage of this package is to provide convenient access to binding prediction methods and perform analysis on the results. There are multiple potential applications.

7.1 Methodology

MHC binding and other prediction methods are implemented by inheriting from a Predictor object. All such classes should at minimum override the predict method for scoring a single sequence. This may wrap methods from other python packages or call command line predictors. For example the TeptitePredictor uses the epitopepredict.teptipe module provided with this package.

The predict method should return a Pandas DataFrame. The predict_sequences method is used for multiple protein sequences contained in a dataframe of sequences in a standard format. This is created from a genbank or fasta file (see examples below). For large numbers of sequences predict_sequences should be called with save=True so that the results are saved as each protein is completed to avoid memory issues, since many alleles might be called for each protein. Results are saved with one file per protein/sequence in csv format.

The results are of the following form and are returned sorted by the score column:

<table>
<thead>
<tr>
<th>peptide</th>
<th>core</th>
<th>pos</th>
<th>score</th>
<th>name</th>
<th>allele</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIFRLMRTNFL</td>
<td>FRLMRTNFL</td>
<td>198</td>
<td>3.4</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>1</td>
</tr>
<tr>
<td>IFRLMRTNFLI</td>
<td>FRLMRTNFL</td>
<td>199</td>
<td>3.4</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>1</td>
</tr>
<tr>
<td>FRLMRTNFLIK</td>
<td>FRLMRTNFL</td>
<td>200</td>
<td>3.4</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>1</td>
</tr>
<tr>
<td>NRFVTLGQQF</td>
<td>FVTLDGQQF</td>
<td>709</td>
<td>2.5</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>4</td>
</tr>
<tr>
<td>RFVTLGQQFY</td>
<td>FVTLDGQQF</td>
<td>710</td>
<td>2.5</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>4</td>
</tr>
<tr>
<td>FVTLDGQQFYW</td>
<td>FVTLDGQQF</td>
<td>711</td>
<td>2.5</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>4</td>
</tr>
<tr>
<td>DSFLMLCLHH</td>
<td>FLMLCLHH</td>
<td>70</td>
<td>2.0</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>7</td>
</tr>
<tr>
<td>SFLLMLCLHH</td>
<td>FLMLCLHH</td>
<td>71</td>
<td>2.0</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>7</td>
</tr>
<tr>
<td>FLMLCLHHAY</td>
<td>FLMLCLHH</td>
<td>72</td>
<td>2.0</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>7</td>
</tr>
<tr>
<td>QGIVRQRVIPV</td>
<td>IVRQRVIPV</td>
<td>32</td>
<td>1.7</td>
<td>ZEBOVgp1</td>
<td>HLA-DRB1*0101</td>
<td>10</td>
</tr>
</tbody>
</table>
where name is the protein identifier from the input file (a locus tag for example) and a score column which will differ between methods. MHC-II methods can be run for varying lengths, with the core usually being the highest scoring in that peptide/n-mer (but not always).

### 7.2 Basics

imports:

```python
import epitopepredict as ep
from epitopepredict import base, sequtils, analysis, plotting
```

create a Predictor object:

```python
# get list of predictors
print base.predictors
['tepitope', 'netmhcipan', 'iedbmhc1', 'iedbmhc2', 'mhcflurry', 'mhcnuggets',
 →'iedbbcell']
p = base.get_predictor('tepitope')
```

get sequence data:

```python
# get data in genbank format into a dataframe
df = sequtils.genbank2Dataframe(genbankfile, cds=True)
# get sequences from fasta file
df = sequtils.fasta2Dataframe(fastafile)
```

run predictions for a protein sequence:

```python
seq = ep.testsequence
label = 'myprot' # optional label for your sequence
p = base.get_predictor('tepitope')
p.predict(sequence=seq, allele='HLA-DRB1*01:01', length=11, name=label)
```

run predictions for multiple proteins:

```python
# run for 2 alleles and save results to savepath
alleles = ["HLA-DRB1*01:01", "HLA-DRB1*03:05"]
p = base.get_predictor('tepitope')
p.predict_proteins(df, length=11, alleles=alleles, save=True, path=savepath)
```

run predictions for a list of peptides:

```python
from epitopepredict import peptutils
seqs = peptutils.create_random_sequences(5000)
p = ep.get_predictor('tepitope')
x = p.predict_peptides(seqs, alleles=alleles)
```

run with multiple cpus:

```python
x = p.predict_peptides(seqs, alleles=alleles, cpus=4)
```

load previous results into a predictor:

```python
p.load(path=path) # where path stores csv files for multiple proteins
p.load(filename=file) # where file is a csv formatted file of prediction results (can be 1 or more proteins)
```
7.3 Analysis

get all the binders using the current data loaded into the predictor:

```python
#default is to use percentile cutoff per allele, returns a dataframe
p.get_binders(cutoff=.95)
```

get binders for only one protein by top median rank:

```python
p.get_binders(name=name, cutoff=10, cutoff_method='rank')
```

get all promiscuous binders, returns a dataframe:

```python
pb = p.promiscuous_binders(n=2, cutoff=.95)
#same using score cutoff
pb = p.promiscuous_binders(n=2, cutoff_method='score', cutoff=500)
```

find clusters of binders in these results:

```python
cl = analysis.find_clusters(b, method, dist=9, minsize=3)
```
8.1 About

It is now known that tumors elicit adaptive immune responses and that the antigens driving effective T cell response are generated from somatically mutated genes. Cancer vaccines and adoptive T cell therapy therefore depends on identification of these patient-specific potential neo-epitopes that might be targeted. This is achieved by applying whole exome sequencing of matched cancer and normal tissues, usually along with RNA-seq quantification of tumour gene expression. Variant calling is then performed, the new mutated peptides extracted and potential epitopes predicted. The computational aspect is broadly outlined below. There are several other software pipelines designed for this task, notably pvactool and MuPeXI. Users are encouraged to try at least one of these also.
8.2 Method

The program currently accepts vcf or maf files that have been output from a variant calling program. In future this step will be added so that the user can provide raw reads. The vcf/maf file is processed using the varcode python library for variant effect prediction to estimate potential mutated coding sequence. The resulting mutated sequence regions are broken up into peptides which are then filtered for similarity to the self proteome. MHC binding predictions are then performed on the remainder.

8.3 Usage and Configuration

The pipeline is run via the same command line tool using epitopepredict. This uses the same text configuration file to provide the inputs and settings. Settings specific to neoepitope prediction are in the [neoepitope] section. These are explained below. Running the tool is then as simple as calling this command:

```
epitopepredict -c <yourfilename>.conf -n
```

You can test the neoepitope pipeline after installing by running:

```
epitopepredict -n -t
```

This will check that the human reference genome is available. (Note for users running the snap package: these files will be placed in your home directory usually under /home/user/snap/epitopepredict/x1.cache/pyensembl/. If you uninstall the package you can also delete this folder to clear space).
neopredict accepts one or more vcf or maf files which have been created from a variant calling program. If more than one file, they should be comma separated. These file names specified in the [neoepitope] of the configuration file:

```
[neoepitope]
vcf_files =
maf_files =
```

The remaining options not in this section are in the other sections covered in the command line interface section of the docs. For example you can specify the prediction algorithm, alleles and length of peptides in those sections.

### 8.4 References

Human or animal MHC allele names have a unique number corresponding to up to four sets of digits separated by colons. The length of the allele designation is dependent on the sequence of the allele and that of its nearest relative. All alleles receive at least a four digit name, which corresponds to the first two sets of digits, longer names are only assigned when necessary. See the full explanation at http://hla.alleles.org/nomenclature/naming.html.

For epitopemap, you can see in the job submission form that we only use the four digit names e.g. HLA-A*68:02. The binding prediction algorithms are not designed to distinguish on a finer level. The four digit names are usually enough for most purposes.

**Examples**

MHC-I allele (using the IEDB tools):

| HLA-A*68:02 |

For MHC-II alleles we use the following format:

| HLA-DRB1*01:01 |

Some methods will work if you leave out the colon separator but not all, so it’s best to use the standard naming scheme.

**References**

- [http://hla.alleles.org/nomenclature](http://hla.alleles.org/nomenclature)
- [https://www.ebi.ac.uk/ipd/mhc/](https://www.ebi.ac.uk/ipd/mhc/)
The command line interface currently accepts protein sequences as fasta or genbank files. Users will probably be familiar with these formats anyway if they are using them. A few useful tips are provided here:

- try to use sensible names in fasta file headers as they are used as identifiers for the results. When you get fasta files from some sources they can have very long headers like this:

```
>lcl|NC_001802.1_prot_NP_057849.4_1 [gene=gag-pol] [locus_tag=HIV1gp1] [db_xref=GeneID:155348] [protein=Gag-Pol] [exception=ribosomal slippage] [protein_id=NP_057849.4] [location=join(336..1637,1637..4642)] [gbkey=CDS]
```

By default the text before the first space is used as the identifier for each protein so that should be unique. In this case it will be lcl|NC_001802.1_prot_NP_057849.4_1. You can also include an option called `fasta_header_sep` in the configuration file that will split the fasta name with another symbol as well, in this way you can shorten the names further, but they should still be unique.

- only CDS (coding sequence) features are used from genbank files, as obviously these are the ones with sequences

- make sure the `/translation` qualifier is present in the features of the genbank file. Some files might not have it and therefore no sequence is present. A typical genbank feature looks like this:

```
CDS 360172..360507
/locus_tag="1mo0332"
/experiment="EXISTENCE:[PMID:19448609]"
/note="1mo0332"
/codon_start=1
/transl_table=11
/product="hypothetical protein"
/protein_id="NP_463862.1"
/db_xref="GeneID:987567"
/translation="MIYYICALYTFIALVSFGFSLDALLKSRKVNGDALINAKYAVSRSLSLIVALGLFIFKSDAFLVALSLVMIGAQLFDGIIGIKISTFKTVGPLLTAVGNVIMLILFLTI"
```
11.1 epitopepredict package

11.1.1 Submodules

11.1.2 epitopepredict.analysis module

epitopepredict analysis methods for workflows Created September 2013 Copyright (C) Damien Farrell

`epitopepredict.analysis.align_blast_results(df, aln=None, idkey='accession', productkey='definition')`

Get gapped alignment from blast results using muscle aligner.

`epitopepredict.analysis.alignment_to_dataframe(aln)`

`epitopepredict.analysis.create_nmers(df, genome, length=20, seqkey='translation', key='nmer', how='split', margin=0)`

Get n-mer peptide surrounding a set of sequences using the host protein sequence. 
: param df: input dataframe with sequence name and start/end coordinates 
: param genome: genome dataframe with host sequences 
: param length: length of nmer to return 
: param seqkey: column name of sequence to be processed 
: param how: method to create the n-mer, 'split' will try to split up

the sequence into overlapping n-mes of length is larger than size

**Parameters margin** – do not split sequences below length+margin

**Returns** pandas Series with nmer values

`epitopepredict.analysis.dbscan(B=None, x=None, dist=7, minsize=4)`

Use dbscan algorithm to cluster binder positions

`epitopepredict.analysis.epitope_conservation(peptides, alnrows=None, proteinseq=None, blastresult=None, blastdb=None, perc_ident=50, equery='srcdb_refseq[Properties]')`
Find and visualise conserved peptides in a set of aligned sequences.

- **peptides**: a list of peptides/epitopes
- **alnrows**: a dataframe of previously aligned sequences e.g. custom strains
- **proteinseq**: a sequence to blast and get an alignment for
- **blastresult**: a file of saved blast results in plain csv format
- **query**: blast query string

**Returns** Matrix of 0 or 1 for conservation for each epitope/protein variant

```python
epitopepredict.analysis.find_clusters(binders, dist=None, min_binders=2, min_size=12, max_size=50, genome=None, colname='peptide')
```

Get clusters of binders for a set of binders. 

- **binders**: dataframe of binders
- **dist**: distance over which to apply clustering
- **min_binders**: minimum binders to be considered a cluster
- **min_size**: smallest cluster length to return
- **max_size**: largest cluster length to return
- **genome**: name for cluster sequence column

**Returns** a pandas Series with the new n-mers (may be longer than the initial dataframe if splitting)

```python
epitopepredict.analysis.find_conserved_peptide(peptide, recs)
```

Find sequences where a peptide is conserved

```python
epitopepredict.analysis.find_conserved_sequences(seqs, alnrows)
```

Find if sub-sequences are conserved in given set of aligned sequences

- **seqs**: a list of sequences to find
- **alnrows**: a dataframe of aligned protein sequences

**Returns** a pandas DataFrame of 1 or 0 values for each protein/search sequence

```python
epitopepredict.analysis.get_AAcontent(df, colname, amino_acids=None)
```

Amino acid composition for dataframe with sequences

```python
epitopepredict.analysis.get_orthologs(seq, db=None, expect=1, hitlist_size=400, query=None, email=None)
```

Fetch orthologous sequences using remote or local blast and return the records as a dataframe.

- **seq**: sequence to blast
- **db**: the name of a local blast db
- **expect**: expect value
- **query**: Entrez Gene Advanced Search options,


**Returns** blast results in a pandas dataframe

```python
epitopepredict.analysis.get_overlaps(df1, df2, label='overlap', how='inside')
```

Overlaps for 2 sets of sequences where the positions in host sequence are stored in each dataframe as ‘start’ and ‘end’ columns

- **df1**: first set of sequences, a pandas dataframe with columns called start/end or pos

**Parameters**

- **df2**: second set of sequences
- **label**: label for overlaps column
- **how**: may be ‘any’ or ‘inside’

**Returns** First DataFrame with no. of overlaps stored in a new column

```python
epitopepredict.analysis.get_seqdepot(seq)
```

Fetch seqdepot annotation for sequence

```python
epitopepredict.analysis.get_species_name(s)
```

Find [species name] in blast result definition

```python
epitopepredict.analysis.isoelectric_point(df)
```

Amino acid composition for dataframe with sequences
epitopepredict.analysis.net_charge\(df, \text{colname}\)  
Net peptide charge for dataframe with sequences

epitopepredict.analysis.peptide_properties\(df, \text{colname}='\text{peptide}')\)  
Find hydrophobicity and net charge for peptides

epitopepredict.analysis.prediction_coverage\(\text{expdata, binders, key='sequence', perc=50, verbose=False}\)  
Determine hit rate of predictions in experimental data by finding how many top peptides are needed to cover % positives :param \text{expdata}: dataframe of experimental data with peptide sequence and name column :param \text{binders}: dataframe of ranked binders created from predictor :param \text{key}: column name in \text{expdata} for sequence  
Returns fraction of predicted binders required to find perc total response

epitopepredict.analysis.randomized_lists\(df, n=94, seed=8, filename='\text{peptide_lists}')\)  
Return a randomized lists of sequences from a dataframe. Used for providing peptide lists for assaying etc.

epitopepredict.analysis.signalP\(\text{infile=None, genome=None}\)  
Get signal peptide predictions

epitopepredict.analysis.test()  

epitopepredict.analysis.test_conservation\(\text{label, gname}\)  
Conservation analysis

epitopepredict.analysis.test_features()  
test feature handling

epitopepredict.analysis.testrun\(\text{gname}\)  

epitopepredict.analysis.tmhmm\(\text{fastafile=None, infile=None}\)  
Get TMHMM predictions :param \text{fastafile}: fasta input file to run :param \text{infile}: text file with tmhmm prediction output

11.1.3 epitopepredict.app module

MHC prediction command line script Created March 2016 Copyright (C) Damien Farrell

class epitopepredict.app.WorkFlow\(\text{opts}={}\)  
Bases: object

Class for implementing a prediction workflow from a set of options

analysis()  
Do analysis of prediction results.

combine\(\text{data}\)  
Combine peptide binders present in all methods.

get_summary\(\text{P, pb, seqs, clusters=None}\)  
Get summary table for sequence based predictions.

plot_results()  
Plot results of predictions

run()  
Run prediction workflow

setup()  
Setup main parameters

epitopepredict.app.add_path()  
Add home dir to path for accessing tools from a snap
epitopepredict Documentation

epitopepredict.app.check_iedb_method(method)
epitopepredict.app.check_iedbmhc1_path()
epitopepredict.app.check_iedbmhc2_path()
epitopepredict.app.check_installed()
  Check which predictors can be used
epitopepredict.app.check_mhcl_length(l)
epitopepredict.app.get_alleles(f)
  Get input alleles as text file or list
epitopepredict.app.get_sequences(filename, header_sep=None)
  Determine file type and get sequences
epitopepredict.app.iedb_checks(method)
epitopepredict.app.list_alleles()
epitopepredict.app.main()
  Run the application
epitopepredict.app.print_help()
epitopepredict.app.read_names(filename)
  read plain text file of items
epitopepredict.app.set_defaults(d)
  Override default paths if provided in conf
epitopepredict.app.show_predictors()
epitopepredict.app.show_preset_alleles()
epitopepredict.app.test_binary()
epitopepredict.app.test_run()
  Test run for a sample file.

11.1.4 epitopepredict.base module

MHC prediction base module for core classes Created November 2013 Copyright (C) Damien Farrell

class epitopepredict.base.BasicMHCIPredictor(data=None, scoring=None)
  Bases: epitopepredict.base.Predictor
  Built-in basic MHC-I predictor. Should be used as a fallback if no other predictors available.
  
  check_install()

get_alleles()
  Get available alleles - override

predict(peptides, allele='HLA-A*01:01', name='temp', **kwargs)
  Encode and predict peptides with daved regressor

prepare_data(df, name, allele)
  Put raw prediction data into DataFrame and rank, override for custom processing. Can be overridden for custom data.

supported_lengths()
  Return supported peptide lengths
class epitopepredict.base.DataFrameIterator(files)
    Simple iterator to get dataframes from a path out of memory

    next()

class epitopepredict.base.DummyPredictor(data=None, scoring=None)
    Bases: epitopepredict.base.Predictor
    Returns random scores. Used for testing

    predict(peptides, allele='HLA-A*01:01', name='temp', **kwargs)
        Does the actual scoring of a sequence. Should be overridden. Should return a pandas DataFrame

class epitopepredict.base.IEDBMHCIIPredictor(data=None)
    Bases: epitopepredict.base.Predictor
    Using IEDB MHC-II method, requires tools to be installed locally

    check_install()
    get_alleles()
    predict(sequence=None, peptides=None, length=15, overlap=None, show_cmd=False, allele='HLA-DRB1*01:01', method='IEDB_recommended', name='', **kwargs)
        Use IEDB MHC-II python module to get predictions. Requires that the IEDB MHC-II tools are installed locally. A sequence argument is provided since the cmd line only accepts whole sequence to be fragmented.

    prepare_data(rows, name)
        Read data from raw output

class epitopepredict.base.IEDBMHCIPredictor(data=None, method='IEDB_recommended')
    Bases: epitopepredict.base.Predictor
    Using IEDB tools method, requires iedb-mhc1 tools. Tested with version 2.17

    check_install()
    get_allele_data()

    get_alleles()
        Get available alleles from model_list file and convert to standard names

    predict(sequence=None, peptides=None, length=11, overlap=1, allele='HLA-A*01:01', name='', method=None, show_cmd=False, **kwargs)
        Use IEDB MHCI python module to get predictions. Requires that the IEDB MHC tools are installed locally

        Returns pandas dataframe

    prepare_data(rows, name)
        Prepare data from results

class epitopepredict.base.MHCFlurryPredictor(data=None, **kwargs)
    Bases: epitopepredict.base.Predictor
    Predictor using MHCFlurry for MHC-I predictions. Requires you to install the python package mhcflurry with dependencies. see https://github.com/hammerlab/mhcflurry

    check_install()

    convert_allele_name(r)

    get_alleles()
        Get available alleles - override
predict (peptides=None, overlap=1, allele='HLA-A0101', name='', **kwargs)
    Uses mhcllury python classes for prediction
prepare_data (df, name, allele)
    Post process dataframe to alter some column names

class epitopepredict.base.MHCNuggetsPredictor (data=None)
    Bases: epitopepredict.base.Predictor
    Predictor using MHCNuggets for MHC-I predictions. Requires you to install the package locally from https://github.com/KarchinLab/mhcnuggets see https://www.biorxiv.org/content/early/2017/07/27/154757
    get_alleles ()
        Get available alleles - override
    predict (peptides=None, overlap=1, allele='HLA-A01:01', name='temp', **kwargs)
        Uses cmd line call to mhcnuggets.
    prepare_data (df, name, allele)
        Get result into dataframe
write_seqs (peptides)

class epitopepredict.base.NetMHCIIPanPredictor (data=None)
    Bases: epitopepredict.base.Predictor
    netMHCIIpan predictor
    allele_mapping (allele)
    check_install ()
    convert_allele_name (a)
        Convert allele names to internally used form
    get_alleles ()
        Get available alleles
    predict (peptides, allele='HLA-DRB1*0101', name='temp', pseudosequence=None, show_cmd=False, **kwargs)
        Call netMHCIIpan command line.
    prepare_data (df, name)
        Prepare netmhciipan results as a dataframe
read_result (res)
    Read raw results from netMHCIIpan output

class epitopepredict.base.NetMHCPanPredictor (data=None, scoring='affinity')
    Bases: epitopepredict.base.Predictor
    netMHCPan 4.0 predictor see http://www.cbs.dtu.dk/services/NetMHCPan/ Default scoring is affinity predictions. To get newer scoring behaviour pass scoring='ligand' to constructor.
    check_install ()
    convert_allele_name (a)
        Convert allele names to internally used form
    get_alleles ()
        Get available alleles
    predict (peptides, allele='HLA-A*01:01', name='temp', pseudosequence=None, show_cmd=False, **kwargs)
        Call netMHCPan command line.
prepare_data \((df, name)\)
Prepare netmhcpan results

read_result \((temp)\)
Read raw results from netMHCpan output

class epitopepredict.base.Predictor \((data=None)\)
Base class to handle generic predictor methods, usually these will wrap methods from other modules and/or call command line predictors. Subclass for specific functionality

allele_summary \((cutoff=5)\)
Allele based summary

class check_alleles \((alleles)\)

class check_install ()

class cleanup ()
Remove temp files from predictions

evaluate \((df, key, value, operator='<')\)
Evaluate binders less than or greater than a cutoff. This method is called by all predictors to get binders

format_row \((x)\)

global_rank
Add a ranking column according to scorekey

global_ranking \((df)\)
Add a ranking column according to scorekey

global_scores \((allele)\)
Return peptides and scores only for an allele

global_unique_cores \((binders=False)\)
Get only unique cores

load \((path=None, names=None, compression='infer', file_limit=None)\)
Load results from path or single file. See results_from_csv for args.

global_rank
Add a ranking column according to scorekey

global_scores \((allele)\)
Return peptides and scores only for an allele

global_unique_cores \((binders=False)\)
Get only unique cores

load \((path=None, names=None, compression='infer', file_limit=None)\)
Load results from path or single file. See results_from_csv for args.
plot (name, **kwargs)
Use module level plotting.mpl_plot_tracks method for predictor plot :param name: :param n: min no. of alleles to be visible :param perc: percentile cutoff for score :param cutoff_method: method to use for cutoffs

predict (sequence=None, peptides=None, length=9, overlap=1, allele='', name='')
Does the actual scoring of a sequence. Should be overridden. Should return a pandas DataFrame

predict_peptides (peptides, cpus=1, path=None, overwrite=True, name=None, **kwargs)
Predict a set of individual peptides without splitting them. This is a wrapper for _predict_peptides to allow multiprocessing. :param peptides: list of peptides :param alleles: list of alleles to predict :param drop_columns: only keep default columns

Returns dataframe with results

predict_proteins (args, **kwargs)
Alias to predict_sequences

predict_sequences (recs, alleles=[], path=None, verbose=False, names=None, key='locus_tag', seqkey='translation', cpus=1, **kwargs)
Get predictions for a set of proteins over multiple alleles that allows running in parallel using the cpus parameter. This is a wrapper for _predictSequences with the same args.

Args:
recs: list or dataframe with sequences cpus: number of processors key: seq/protein name key seqkey: key for sequence column

Returns:
a dataframe of predictions over multiple proteins

prepare_data (result, name, allele)
Put raw prediction data into DataFrame and rank, override for custom processing. Can be overridden for custom data.

print_heading ()

promiscuous_binders (binders=None, name=None, cutoff=0.95, cutoff_method='default', n=1, unique_core=True, **kwargs)
Use params for getbinders if no binders provided? :param binders: can provide a precalculated list of binders :param name: specific protein, optional :param value: to pass to get_binders :param cutoff_method: 'default', 'score' or 'rank' :param cutoff: percentile cutoff for get_binders :param n: min number of alleles :param unique_core: removes peptides with duplicate cores and picks the most :param promiscuous and highest ranked, used for mhc-II predictions:

Returns a pandas dataframe

protein_summary ()

proteins ()

ranked_binders (names=None, how='median', cutoff=None)
Get the median/mean rank of each binder over all alleles. :param names: list of protein names, otherwise all current data used :param how: method to use for rank selection, 'median' (default), 'best' or 'mean': :param cutoff: apply a rank cutoff if we want to filter (optional)

reshape (name=None)
Return pivoted data over alleles for summary use

save (prefix='_', filename=None, compression=None)
Save all current predictions dataframe with some metadata :param prefix: if writing to a path, the prefix name :param filename: if saving all to a single file :param compression: a string representing the compression to use, :param allowed values are 'gzip', 'bz2', 'xz':

save_msgpack (filename=None)
Save as msgpack format - experimental
**seqs_to_dataframe** *(seqs)*

**summarize** *
Summarise currently loaded data

**supported_lengths** *
Return supported peptide lengths

**class** **epitopepredict.base.TEpitopePredictor** *(data=None, **kwargs)*
**Bases:** **epitopepredict.base.Predictor**

Predictor using TepitopePan QM method

**check_alleles** *(alleles)*

**get_alleles** *
Get available alleles - override

**predict** *(peptides=None, length=9, overlap=1, allele='HLA-DRB1*0101', name='', pseudosequence=None, **kwargs)*
Does the actual scoring of a sequence. Should be overriden. Should return a pandas DataFrame

**supported_lengths** *
Return supported peptide lengths

**epitopepredict.base.check_snap** *
Check if inside a snap

**epitopepredict.base.clean_sequence** *(seq)*
Clean a sequence of invalid characters before prediction

**epitopepredict.base.compare_predictors** *(p1, p2, by='allele', cutoff=5, n=2)*
Compare predictions from 2 different predictors. **param** p1, p2: predictors with prediction results for the same **param** set of sequences and alleles: **param** by: how to group the correlation plots

**epitopepredict.base.first** *(x)*

**epitopepredict.base.get_coords** *(df)*
Get start end coords from position and length of peptides

**epitopepredict.base.get_dqp_list** *(a)*
Get DRB list in standard format

**epitopepredict.base.get_drb_list** *(a)*
Get DRB list in standard format

**epitopepredict.base.get_filenames** *(path, names=None, file_limit=None)*

**epitopepredict.base.get_iedb_request** *(seq, alleles='HLA-DRB1*01:01', method='consensus3')*

**epitopepredict.base.get_length** *(data)*
Get peptide length of a dataframe of predictions

**epitopepredict.base.get_nearest** *(df)*
Get nearest binder

**epitopepredict.base.get_overlapping** *(index, s, length=9, cutoff=25)*
Get all mutually overlapping kmers within a cutoff area

**epitopepredict.base.get_pos** *(x)*

**epitopepredict.base.get_predictor** *(name='tepitope', **kwargs)*
Get a predictor object using it's name. Valid predictor names are held in the predictors attribute.

11.1. epitopepredict package
epitopepredict Documentation

epitopepredict.base.get_predictor_classes()
    Get predictor classes in this module.

epitopepredict.base.get_preset_alleles(name)

epitopepredict.base.get_quantiles(predictor)
    Get quantile score values per allele in set of predictions. Used for making pre-defined cutoffs. :param predictor: predictor with set of predictions

epitopepredict.base.get_sequence(seqfile)
    Get sequence from fasta file

epitopepredict.base.get_standard_mhc1(name)
    Taken from iedb mhc1 utils.py

epitopepredict.base.get_standard_mhc2(x)

epitopepredict.base.plot_summary_heatmap(p, kind='default', name=None)
    Plot heatmap of binders using summary dataframe.

epitopepredict.base.predict_peptides_worker(P, recs, kwargs)

epitopepredict.base.predict_proteins_worker(P, recs, kwargs)

epitopepredict.base.protein_summary(pred, peptides, name)
    formatted protein summary table

epitopepredict.base.read_defaults()
    Get some global settings such as program paths from config file

epitopepredict.base.reshape_data(pred, peptides=None, name=None, values='score')
    Create summary table per binder/allele with cutoffs applied. :param pred: predictor with data :param cutoff: percentile cutoff :param n: number of alleles

epitopepredict.base.results_from_csv(path=None, names=None, compression='infer', file_limit=None)
    Load results for multiple csv files in a folder or a single file. :param path: name of a csv file or directory with one or more csv files :param names: names of proteins to load :param file_limit: limit to load only the this number of proteins

epitopepredict.base.seq_from_binders(df)

epitopepredict.base.set_netmhcpan_cmd(path=None)
    Setup the netmhcpan command to point directly to the binary. This is a workaround for running inside snaps. Avoids using the tcsh script.

epitopepredict.base.split_peptides(df, length=9, seqkey='sequence', newcol='peptide')
    Split sequences in a dataframe into peptide fragments

epitopepredict.base.summarize(data)
    Summarise prediction data

epitopepredict.base.summarize_by_protein(pred, pb)
    Heatmaps or tables of binders per protein/allele

epitopepredict.base.write_fasta(sequences, id=None, filename='tempseq.fa')

11.1.5 epitopepredict.cluster module

11.1.6 epitopepredict.config module

epitopepredict config Created March 2016 Copyright (C) Damien Farrell
epitopepredict.config.check_options(opts)
    Check for missing default options in dict. Meant to handle incomplete config files

epitopepredict.config.create_config_parser_from_dict(data=None, sections=['base', 'iedbtools'], **kwargs)
    Helper method to create a ConfigParser from a dict of the form shown in baseoptions

epitopepredict.config.get_options(cp)
    Makes sure boolean opts are parsed

epitopepredict.config.parse_config(conffile=None)
    Parse a configparser file

epitopepredict.config.print_options(options)
    Print option key/value pairs

epitopepredict.config.write_config(conffile='default.conf', defaults={})
    Write a default config file

epitopepredict.config.write_default_config()
    Write a default config to users .config folder. Used to add global settings.

11.1.7 epitopepredict.neo module

Command line script for neo epitope prediction Created March 2018 Copyright (C) Damien Farrell

class epitopepredict.neo.NeoEpitopeWorkflow(opts={})
    Bases: object
    Class for implementing a neo epitope workflow.
    combine_samples(labels)
        Put peptides from multiple files in one table
    get_file_labels(files)
    run()
        Run workflow for multiple samples and prediction methods.
    setup()
        Setup main parameters

epitopepredict.neo.anchor_mutated(x)

epitopepredict.neo.check_ensembl(release='75')
    Check pyensembl ref genome cached. Needed for running in snap

epitopepredict.neo.check_imports()

epitopepredict.neo.combine_wt_scores(x, y, key)
    Combine mutant peptide and matching wt/self binding scores from a set of predictions. Assumes both
dataframes were run with the same alleles. :param x,y: pandas dataframes with matching prediction results
    :param key:

epitopepredict.neo.dataframe_to_vcf(df, outfile)
    Write a dataframe of variants to a simple vcf file. Dataframe requires the following columns:
    #CHROM,POS,ID,REF,ALT

epitopepredict.neo.efffects_to_dataframe(effects)

epitopepredict.neo.efffects_to_pickle(effects, filename)
    serialize variant effects collections
epitopepredict.neo.fetch_ensembl_release(path=None, release='75')
    Get pyensembl genome files

epitopepredict.neo.find_matches(df, blastdb, cpus=4, verbose=False)
    Get similarity measures for peptides to a self proteome. Does a local blast to the proteome and finds most similar
    matches. These can then be scored. :param df: dataframe of peptides :param blastdb: path to protein blastdb

    Returns  'sseq', 'mismatch'
    Return type dataframe with extra columns

epitopepredict.neo.get_alleles(f)
    Get input alleles

epitopepredict.neo.get_closest_match(x)
    Create columns with closest matching peptide. If no wt peptide use self match. vector method

epitopepredict.neo.get_closest_matches(df, verbose=False, cpus=1)
    Find peptide similarity metrics

epitopepredict.neo.get_mutation_sequences(variants=None, effects=None, reference=None, peptides=True, drop_duplicates=True, length=11, verbose=False)
    Get mutant proteins or peptide fragments from vcf or maf file. :param variants: varcode variant collection
    :param effects: non-synonmymous effects, alternative to variants :param peptides: get peptide fragments around
    mutation

    Returns  pandas dataframe with mutated peptide sequence and source information

epitopepredict.neo.get_variant_class(effect)

epitopepredict.neo.get_variants_effects(variants, gene_expression_dict=None, verbose=False)
    Get all effects from a list of variants. :returns: list of varcode variant effect objects

epitopepredict.neo.load_variants(vcf_file=None, maf_file=None, max_variants=None)
    Load variants from vcf file

epitopepredict.neo.make_blastdb(url, name=None, filename=None, overwrite=False)
    Download protein sequences and a make blast db. Uses datacache module.

epitopepredict.neo.make_human_blastdb()
    Human proteome blastdb

epitopepredict.neo.make_virus_blastdb()
    Human virus blastdb

epitopepredict.neo.pbmec_score(seq1, seq2)
    Score with PBMEC matrix

epitopepredict.neo.peptides_from_effect(ef, length=11, peptides=True, verbose=False)
    Get mutated peptides from a single effect object. :returns: dataframe with peptides and variant info

epitopepredict.neo.plot_variant_summary(data)

epitopepredict.neo.predict_binding(df, predictor='netmhcpan', alleles=[], verbose=False, cpus=1, cutoff=0.95, cutoff_method='default')
    Predict binding scores for mutated and wt peptides (if present) from supplied variants. :param df: pandas
    dataframe with peptide sequences, requires at least 2 columns

    'peptide' - the mutant peptide 'wt' - a corresponding wild type peptide

    Parameters
- data could be generated from `get_mutant_sequences` or from an external program (this) -
- predictor – mhc binding prediction method
- alleles – list of alleles

**Returns** dataframe with mutant and wt binding scores for all alleles

```
epitopepredict.neo.print_help()
```

```
epitopepredict.neo.read_names(filename)
```

```
epitopepredict.neo.run_vep(vcf_file, out_format='vcf', assembly='GRCh38', cpus=4,
path=None)
```

Run ensembl VEP on a vcf file for use with pvacseq. see https://www.ensembl.org/info/docs/tools/vep/script/index.html

```
epitopepredict.neo.score_peptides(df, rf=None)
```

Score peptides with a classifier. Returns a prediction probability.

```
epitopepredict.neo.self_matches(df, **kwargs)
```

```
epitopepredict.neo.self_similarity(x, matrix='blosum62')
```

```
epitopepredict.neo.show_predictors()
```

```
epitopepredict.neo.summary_plots(df)
```

summary plots for testing results

```
epitopepredict.neo.test_run()
```

Test run for sample vcf file

```
epitopepredict.neo.varcode_test()
```

```
epitopepredict.neo.variants_from_csv(csv_file, sample_id=None, reference=None)
```

Variants from csv file. :param csv_file: csv file with following column names:
- chromosome, position, reference_allele, alt_allele, gene_name, transcript_id, sample_id

**Parameters**

- **sample_id** – if provided, select variants only for this id
- **reference** – ref genome used for variant calling

```
epitopepredict.neo.virus_matches(df, **kwargs)
```

```
epitopepredict.neo.virus_similarity(x, matrix='blosum62')
```

```
epitopepredict.neo.wt_similarity(x, matrix='blosum62')
```

## 11.1.8 epitopepredict.peptutils module

Module implementing peptide sequence/structure utilities. Created March 2013 Copyright (C) Damien Farrell

```
epitopepredict.peptutils.compare_anchor_positions(x1, x2)
```

Check if anchor positions in 9-mers are mutated

```
epitopepredict.peptutils.create_fragments(profile=None, seq=None, length=9, overlap=1,
quiet=True)
```

generate peptide fragments from a sequence
epitopepredict.peptutils.create_random_peptides (size=100, length=9)  
Create random peptide structures of given length

epitopepredict.peptutils.create_random_sequences (size=100, length=9)  
Create library of all possible peptides given length

epitopepredict.peptutils.get_AAfraction (seq, amino_acids=None)  
Get fraction of given amino acids in a sequence

epitopepredict.peptutils.get_AAsubstitutions (template)  
Get all the possible sequences from substituting every AA into the given sequence at each position. This gives a total of

\[19 \text{aa} \times n \text{ positions}\]

epitopepredict.peptutils.get_all_fragments (exp, length=11)  
epitopepredict.peptutils.get_fragments (seq=None, overlap=1, length=11, **kwargs)  
Generate peptide fragments from a sequence. :returns: dataframe of peptides with position column.

epitopepredict.peptutils.main ()

epitopepredict.peptutils.net_charge (seq)  
Get net charge of a peptide sequence

11.1.9 epitopepredict.plotting module

MHCpredict plotting Created February 2016 Copyright (C) Damien Farrell

epitopepredict.plotting.binders_to.coords (df)  
Convert binder results to dict of coords for plotting

epitopepredict.plotting.bokeh_pie_chart (df, title="", radius=0.5, width=400, height=400, palette='Spectral')  
Bokeh pie chart

epitopepredict.plotting.bokeh_plot_bar (preds, name=None, allele=None, title="", width=None, height=100, palette='Set1', tools=True, x_range=None)  
Plot bars combining one or more prediction results for a set of peptides in a protein/sequence

epitopepredict.plotting.bokeh_plot_grid (pred, name=None, width=None, palette='Blues', **kwargs)  
Plot heatmap of binding results for a predictor.

epitopepredict.plotting.bokeh_plot_tracks (preds, title="", n=2, name=None, cutoff=5, cutoff_method='default', width=None, height=None, x_range=None, tools=True, palette='Set1', seqdepot=None, exp=None)  
Plot binding predictions as parallel tracks of blocks for each allele. This uses Bokeh. :param title: plot title :param n: min alleles to display :param name: name of protein to show if more than one in data

Returns: a bokeh figure for embedding or displaying in a notebook

epitopepredict.plotting.bokeh_summary_plot (df, savepath=None)  
Summary plot

epitopepredict.plotting.bokeh_test (height=400)

epitopepredict.plotting.bokeh_vbar (x, height=200, title="", color='navy')

epitopepredict.plotting.draw_labels (labels, coords, ax)  
Add labels on axis
epitopepredict.plotting.get_bokeh_colors(palette='Set1')
epitopepredict.plotting.get_seq_from_binders(P, name=None)
    Get sequence from binder data. Probably better to store the sequences in the object?
epitopepredict.plotting.get_seqdepot_annotation(genome, key='pfam27')
    Get seqdepot annotations for a set of proteins in dataframe.
epitopepredict.plotting.plot_bars(P, name, chunks=1, how='median', cutoff=20,
    color='black')
    Bar plots for sequence using median/mean/total scores. :param P: predictor with data :param name: name of
    protein sequence :param chunks: break sequence up into 1 or more chunks :param how: method to calculate
    score bar value :param perc: percentile cutoff to show peptide
epitopepredict.plotting.plot_bcell(plot, pred, height, ax=None)
    Line plot of iedb bcell results
epitopepredict.plotting.plot_binder_map(P, name, values='rank',
    cutoff=20, chunks=1, cmap=None)
    Plot heatmap of binders above a cutoff by rank or score. :param P: predictor object with data :param name:
    name of protein to plot :param values: data column to use for plot data, ‘score’ or ‘rank’ :param cutoff: cutoff if
    using rank as values :param chunks: number of plots to split the sequence into
epitopepredict.plotting.plot_heatmap(df, ax=None, figsize=(6, 6), **kwargs)
    Plot a generic heatmap
epitopepredict.plotting.plot_multiple(preds, names, kind='tracks',
    regions=None, genome=None, **kwargs)
    Plot results for multiple proteins
epitopepredict.plotting.plot_overview(genome, coords=None, cols=2,
    colormap='Paired', legend=True, figsize=None)
    Plot regions of interest in a group of protein sequences. Useful for seeing how your binders/epitopes are dis-
    tributed in a small genome or subset of genes. :param genome: dataframe with protein sequences :param coords:
    a list/dict of tuple lists of the form {protein name: [(start,length)..]} :param cols: number of columns for plot,
    integer
epitopepredict.plotting.plot_regions(coords, ax, color='red', label='',
    alpha=0.6)
    Highlight regions in a prot binder plot
epitopepredict.plotting.plot_seqdepot(association, ax)
    Plot seqdepot annotations - replace with generic plot coords track
epitopepredict.plotting.plot_tracks(preds, name, n=1, cutoff=0.95, cutoff_method='default',
    regions=None, legend=False, colormap='Paired', figsize=None, ax=None, **kwargs)
    Plot binders as bars per allele using matplotlib. :param preds: list of one or more predictors :param name: name
    of protein to plot :param n: number of alleles binder should be found in to be displayed :param cutoff: percentile
cutoff to determine binders to show
epitopepredict.plotting.seqdepot_to_coords(sd, key='pfam27')
    Convert seqdepot annotations to coords for plotting

11.1.10 epitopepredict.sequtils module

Sequence utilities and genome annotation methods Created November 2013 Copyright (C) Damien Farrell
epitopepredict.sequtils.alignment_to_dataframe(aln)
    Sequence alignment to dataframe

11.1. epitopepredict package
**epitopepredict Documentation**

**epitopepredict.sequtils.blast_sequences** *(database, seqs, labels=None, **kwargs)*

Blast a set of sequences to a local or remote blast database: 
- param database: local or remote blast db name
  - ‘nr’, ‘refseq_protein’, ‘pdb’, ‘swissprot’ are valid remote dbs

**Parameters**

- seqs – sequences to query, list of strings or Bio.SeqRecords
- labels – list of id names for sequences, optional but recommended

**Returns**

pandas dataframe with top blast results

**epitopepredict.sequtils.check_tags** *(df)*

Check genbank tags to make sure they are not empty

**epitopepredict.sequtils.clustal_alignment** *(filename=None, seqs=None, command='clustalw')*

Align 2 sequences with clustal

**epitopepredict.sequtils.convert_sequence_format** *(infile, outformat='embl')*

Convert sequence files using SeqIO

**epitopepredict.sequtils.dataframe_to_fasta** *(df, seqkey='translation', idkey='locus_tag', descrkey='description', outfile='out.faa')*

Genbank features to fasta file

**epitopepredict.sequtils.dataframe_to_seqrecords** *(df, seqkey='sequence', idkey='id')*

dataframe to list of Bio.SeqRecord objects

**epitopepredict.sequtils.distance_tree** *(filename=None, seqs=None, ref=None)*

Basic phylogenetic tree for an alignment

**epitopepredict.sequtils.draw_genome_map** *(infile, filename=None)*

Draw whole circular genome

**epitopepredict.sequtils.embl_to_dataframe** *(infile, cds=False)*

**epitopepredict.sequtils.ete_tree** *(aln)*

Tree showing alleles

**epitopepredict.sequtils.fasta_format_from_feature** *(feature)*

Get fasta formatted sequence from a genome feature

**epitopepredict.sequtils.fasta_to_dataframe** *(infile, header_sep=None, key='locus_tag', seqkey='translation')*

Get fasta proteins into dataframe

**epitopepredict.sequtils.features_to_dataframe** *(rec, cds=False)*

Get a genome record from a biopython features object into a dataframe returns a dataframe with a row for each cds/entry.

**epitopepredict.sequtils.fetch_protein_sequences** *(searchterm, filename='found.fa')*

Fetch protein seqs using ncbi esearch and save results to a fasta file. 
- param searchterm: entrez search term
- param filename: fasta file name to save results

**Returns**

sequence records as a dataframe

**epitopepredict.sequtils.find_keyword** *(f)*

Get keyword from a field

**epitopepredict.sequtils.format_alignment** *(aln)*
epitopepredict.sequats.get_genbank_summary\( (df) \)
Genbank dataframe summary

epitopepredict.sequats.get_genbank_to_dataframe\( (in\text{file}, \text{cds=}\text{False, rec\text{s}=}'\text{all}') \)
Get genome records from a genbank file into a dataframe returns a dataframe with a row for each cds/entry

epitopepredict.sequats.get_blast_results\( (\text{filename}) \)
Get blast results into dataframe. Assumes column names from local_blast method. :returns: dataframe

epitopepredict.sequats.get_cds\( (df) \)
Get CDS with translation from genbank dataframe

epitopepredict.sequats.get_featureQualifier\( (f, \text{qualifier}) \)

epitopepredict.sequats.get_genes_by_location\( (\text{genome,feature, within}=20) \)
Gets all features within a given distance of a gene

epitopepredict.sequats.get_identity\( (\text{aln}) \)
Get sequence identity of alignment for overlapping region only

epitopepredict.sequats.get_sequence\( (\text{genome, name}) \)
Get the sequence for a protein in a dataframe with genbank/sequence data

epitopepredict.sequats.get_translation\( (\text{feature, genome, cds=}\text{True}) \)
Check the translation of a cds feature

epitopepredict.sequats.index_genbank_features\( (\text{gb_record, feature_type, qualifier}) \)
Index features by qualifier value for easy access

epitopepredict.sequats.local_blast\( (\text{database, query, output=None, maxseqs=50, evalue=0.001, compress=False, cmd='blastp', cpus=2, show_cmd=False, **kwargs}) \)
Blast a local database. :param database: local blast db name :param query: sequences to query, list of strings or Bio.SeqRecords

Returns pandas dataframe with top blast results

epitopepredict.sequats.muscle_alignment\( (\text{filename=None, seqs=None}) \)
Align 2 sequences with muscle

epitopepredict.sequats.needle_alignment\( (\text{seq1, seq2, outfile='needle.txt'}) \)
Align 2 sequences with needle

epitopepredict.sequats.pairwise_alignment\( (\text{rec1, rec2}) \)

epitopepredict.sequats.remote_blast\( (\text{db, query, maxseqs=50, evalue=0.001, **kwargs}) \)
Remote blastp. :param query: fasta file with sequence to blast :param db: database to use - nr, refseq_protein, pdb, swissprot

epitopepredict.sequats.show_alignment\( (\text{aln, diff=False, offset=0}) \)
Show a sequence alignment

Args: aln: alignment diff: whether to show differences

epitopepredict.sequats.show_alignment_html\( (\text{alnrows, seqs, width=80, fontsize=15, label='name'}) \)
Get html display of sub-sequences on multiple protein alignment. :param alnrows: a dataframe of aligned sequences :param seqs: sub-sequences/epitopes to draw if present :param label: key from dataframe to use as label for sequences

Returns html code
11.1.11 epitopepredict.tepitope module


epitopepredict.tepitope.allelenumber(x)
epitopepredict.tepitope.benchmark()
epitopepredict.tepitope.compare(file1, file2, alnindex, reduced=True)
  All vs all for 2 sets of sequence files
epitopepredict.tepitope.compare_alleles(alleles1, alleles2, alnindex, reduced=True, cutoff=0.25, matrix=None, matrix_name='blosum62')
  Compare 2 sets of alleles for pseudo-seq distances
epitopepredict.tepitope.compare_ref(query1, query2, ref, alnindex)
  Compare different alleles distances to reference
epitopepredict.tepitope.compare_tepitope_alleles(alnindex)
  Compare a set of alleles to Tepitope library HLAs
epitopepredict.tepitope.convert_allele_names(seqfile)
  Convert long IPD names to common form. :param fasta sequence file:
    Returns new list of seqrecords
epitopepredict.tepitope.create_virtual_pssm(allele)
  Create virtual matrix from pickpocket profile weights
epitopepredict.tepitope.generate_pssm(expdata)
  Create pssm for known binding data given a set of n-mers and binding score
epitopepredict.tepitope.get_allele_pocket_sequences(allele)
  Convenience for getting an allele pocket aas
epitopepredict.tepitope.get_alleles()
  Get all alleles covered by this method.
epitopepredict.tepitope.get_matrix(name)
epitopepredict.tepitope.get_pocket_positions()
epitopepredict.tepitope.get_pockets_pseudo_sequence(query, offset=28)
  Get pockets pseudo-seq from sequence and pocket residues. :param query: query sequence :param offset: seq numbering offset of alignment numbering to pickpocket :param residue values:
epitopepredict.tepitope.get_pseudo_sequence(query, positions=None, offset=28)
  Get non redundant pseudo-sequence for a query. Assumes input is a sequence from alignment of MHC genes.
epitopepredict.tepitope.get_pssm_score(seq, pssm)
  Get sequence score for a given pssm
epitopepredict.tepitope.get_pssms()
  Get tepitope pssm data
epitopepredict.tepitope.get_scores(pssm, sequence=None, peptides=None, length=11, overlap=1)
  Score multiple fragments of a sequence in separate fragments
Get distances between a query and set of ref pseudo-seqs

Derive weights for a query allele using pickpocket method. This uses the pocket pseudosequences to determine similarity to the reference. This relies on the DRB alignment present in the tepitope folder.

Parameters
- \texttt{pos} – pocket position
- \texttt{allele} – query allele

Returns set of weights for library alleles at this position

Reduce alleles to repr set based on names

Score a single sequence in 9-mer frames

Test to show the pocket residues in a pdb structure

Similarity for pseudosequences using a substitution matrix.

Returns a similarity value normalized to matrix

11.1.12 epitopepredict/tests module

MHC prediction unit tests Created September 2015 Copyright (C) Damien Farrell

Basic tests for predictor

Hook method for setting up the test fixture before exercising it.

Test fasta predictions

Test genbank feature handling

iedbmhc1 test
test_load()
    Test re-loading predictions

test_mhcflurry()
    Test mhcflurry predictor

test_mhcflurry()
    Test mhcflurry predictor

test_multiproc()

test_netmhcpan()
    netMHCpan test

test_peptide_prediction()

test_tepitope()
    Tepitope test

epitopepredict.tests.run()

11.1.13 epitopepredict.tornado_serve module

epitopepredict server app for viewing results, uses tornado Created Sep 2017 Copyright (C) Damien Farrell

class epitopepredict.tornado_serve.ConfigForm(formdata=None, obj=None, prefix='', locale_code='en_US', **kwargs)
    Bases: wtforms_tornado.form.Form

cpus = <UnboundField(IntegerField, ('cpus',), {'default': 1})>

i = 'HLA-DRB5*0205'

iedbmhc1_path = <UnboundField(TextField, ('iedb MHC-I tools path'),{},())>

iedbmhc2_path = <UnboundField(TextField, ('iedb MHC-II tools path'),{},())>

mhc1_alleles = <UnboundField(SelectMultipleField, ('MHC-I alleles'), {'render_kw': {'class': 'combobox'}, 'choices': ['RT1A', 'RT1A'], 'default': 'RT1A', 'choice_label': 'RT1A')})>

mhc1_length = <UnboundField(IntegerField, ('mhc1 length'), {'default': 11})>

mhc1_presets = <UnboundField(SelectField, ('MHC-I presets'), {'default': '', 'choices': [('mhc1_supertypes', 'mhc1_supertypes'), ('us_caucasion_mhc1', 'us_caucasion_mhc1'), ('us_african_mhc1', 'us_african_mhc1'), ('broad_coverage_mhc1', 'broad_coverage_mhc1')])})>

mhc2_alleles = <UnboundField(SelectMultipleField, ('MHC-II alleles'), {'render_kw': {'class': 'combobox'}, 'choices': ['HLA-DRB5*0203', 'HLA-DRB5*0203', 'HLA-DRB5*0204', 'HLA-DRB5*0204', 'HLA-DRB5*0205', 'HLA-DRB5*0205'], 'default': 'HLA-DRB5*0205', 'choice_label': 'HLA-DRB5*0205')})>

mhc2_length = <UnboundField(IntegerField, ('mhc2 length'), {'default': 15})>

mhc2_presets = <UnboundField(SelectField, ('MHC-II presets'), {'default': '', 'choices': [('mhc2_supertypes', 'mhc2_supertypes'), ('human_common_mhc2', 'human_common_mhc2'), ('bovine_like_mhc2', 'bovine_like_mhc2')])})>

overwrite = <UnboundField(BooleanField, ('overwrite'), {'default': False, 'false_values': set([False, '', 'n'])})>

p1 = iedbmhc1 predictor

p2 = tepitope predictor

path = <UnboundField(TextField, ('output path'), {'default': 'results', 'render_kw': {'class': 'textbox'}, 'validators': [wtforms.validators.DataRequired object])>

pm = [('basicmhc1', 'basicmhc1'), ('tepitope', 'tepitope'), ('netmhciipan', 'netmhciipan'), ('netmhcpan', 'netmhcpan'), ('mhcflurry', 'mhcflurry'), ('mhcnuggets', 'mhcnuggets'), ('iedbmhc1', 'iedbmhc1'), ('iedbmhc2', 'iedbmhc2')]

predictors = <UnboundField(SelectMultipleField, ('predictors'), {'render_kw': {'class': 'combobox'}, 'choices': pm, 'default': []})>

ps1 = [('mhc1_supertypes', 'mhc1_supertypes'), ('us_caucasion_mhc1', 'us_caucasion_mhc1'), ('us_african_mhc1', 'us_african_mhc1'), ('broad_coverage_mhc1', 'broad_coverage_mhc1')]

ps2 = [('mhc2_supertypes', 'mhc2_supertypes'), ('human_common_mhc2', 'human_common_mhc2'), ('bovine_like_mhc2', 'bovine_like_mhc2')]

sequence_file = <UnboundField(TextField, ('sequence file'), {'default': '', 'validators': [wtforms.validators.DataRequired object], _is_seqfile, _exists})>
x = [('HLA-DRB1*1601', 'HLA-DRB1*1601'), ('HLA-DRB1*0101', 'HLA-DRB1*0101'), ('HLA-DRB1*0102', 'HLA-DRB1*0102'), ...
('HLA-DRB5*0203', 'HLA-DRB5*0203'), ('HLA-DRB5*0204', 'HLA-DRB5*0204'), ('HLA-DRB5*0205', 'HLA-DRB5*0205')]

11.1. epitopepredict package
no_data_render(form, msg)

class epitopepredict.tornado_serve.NeoForm(formdata=None, obj=None, prefix='', locale_code='en_US', **kwargs)
Bases: wtforms_tornado.form.Form

i = 'combined'
sample = <UnboundField(SelectField, ('sample',), {'choices': []})>
savepath = <UnboundField(TextField, ('savepath',), {'default': '', 'render_kw': {'class': ''})>
view = <UnboundField(SelectField, ('view',), {'choices': [('final', 'final'), ('combined', 'combined')])>
views = [('final', 'final'), ('combined', 'combined')]

class epitopepredict.tornado_serve.SequenceViewHandler(application, request, **kwargs)
Bases: tornado.web.RequestHandler
Handler for main results page
get()

get_url(args, link='download')
Get url from current args

class epitopepredict.tornado_serve.SummaryForm(formdata=None, obj=None, prefix='', locale_code='en_US', **kwargs)
Bases: wtforms_tornado.form.Form

deletecached = <UnboundField(BooleanField, ('delete cached',), {})>savepath = <UnboundField(TextField, ('savepath',), {'default': 'results'})>
epitopepredict.tornado_serve.dict_to_html(d)
epitopepredict.tornado_serve.exists(message=u'File does not exist')
epitopepredict.tornado_serve.get_args(args, defaults={'cutoff': 0.95, 'cutoff_method': 'default', 'kind': 'tracks', 'n': 2, 'name': '', 'pred': 'epitope', 'savepath': '', 'view': ''})
epitopepredict.tornado_serve.help_msg()
epitopepredict.tornado_serve.is_seqfile(message=u'Wrong format file. Should be fasta or genbank', extensions=None)
epitopepredict.tornado_serve.main(port=8000)
epitopepredict.tornado_serve.str_to_html(s)

11.1.14 epitopepredict.utilities module

Utilities for epitopepredict Created March 2013 Copyright (C) Damien Farrell

epitopepredict.utilities.add_dicts(a, b)
epitopepredict.utilities.compress(filename, remove=False)
Compress a file with gzip
epitopepredict.utilities.copyfile(source, dest, newname=None)
Helper method to copy files
epitopepredict.utilities.copyfiles(path, files)
epitopepredict.utilities.filter_iedb_file(filename, field, search)
    Return filtered iedb data

epitopepredict.utilities.find_filefrom_string(files, string)

epitopepredict.utilities.find_files(path, ext='txt')
    List files in a dir of a specific type

epitopepredict.utilities.find_folders(path)

epitopepredict.utilities.get_sequencefrom_pdb(pdbfile, chain='C', index=0)
    Get AA sequence from PDB

epitopepredict.utilities.get_symmetric_data_frame(m)

epitopepredict.utilities.read_iedb(filename, key='Epitope ID')
    Load iedb peptidic csv file and return dataframe

epitopepredict.utilities.reorder_filenames(files, order)
    reorder filenames by another list order(seqs)

epitopepredict.utilities.rmse(ar1, ar2)
    Mean squared error

epitopepredict.utilities.search_pubmed(term, max_count=100)

epitopepredict.utilities.symmetrize(m, lower=True)
    Return symmetric array

epitopepredict.utilities.test()

epitopepredict.utilities.venndiagram(names, labels, ax=None, colors=('r', 'g', 'b'), **kwargs)
    Plot a venn diagram

11.1.15 epitopepredict.web module

epitopepredict.web.column_to_url(df, field, path)
    Add urls to specified field in a dataframe by prepending the supplied path.

epitopepredict.web.create_bokeh_table(path, name)
    Create table of prediction data

epitopepredict.web.create_figures(preds, name=", kind=‘tracks’, cutoff=5, n=2, cutoff_method=‘default’, **kwargs)
    Get plots of binders for single protein/sequence

epitopepredict.web.create_sequence_html(preds, name=", classes=", **kwargs)

epitopepredict.web.create_widgets()

epitopepredict.web.dataframes_to_html(data, classes=")
    Convert dictionary of dataframes to html tables

epitopepredict.web.dict_to_html(data)

epitopepredict.web.get_alleles(preds)
    get available alleles

epitopepredict.web.get_file_lists(path)
    Get list of available prediction results in the given path. Tries to check for each possible predictor.
epitopepredict.web.get_predictors(path, name=None)
Get a set of predictors under a results path for all or a specific protein.

epitopepredict.web.get_results_info(P)
Info on sequence used for prediction

epitopepredict.web.get_results_tables(path, name=None, promiscuous=True, limit=None, **kwargs)
Get binder results from a results path. :param path: path to results :param name: name of particular protein/sequence :param view: get all binders or just promiscuous

epitopepredict.web.get_sequences(pred)
Get set of sequences from loaded data

epitopepredict.web.get_summary_tables(path, limit=None, **kwargs)
Get binder results summary for all proteins in path. :param path: path to results

epitopepredict.web.sequence_from_peptides(df)
Derive sequence from set of peptides

epitopepredict.web.sequence_to_html_grid(preds, classes=", **kwargs)
Put aligned or multiple identical rows in dataframe and convert to grid of aas as html table

epitopepredict.web.sequences_to_html_table(seqs, classes="")
Convert seqs to html

epitopepredict.web.tabbed_html(items)
Create html for a set of tabbed divs from dict of html code, one for each tab. Uses css classes defined in static/custom.css

epitopepredict.web.test()
CHAPTER 12

Indices and tables

- genindex
- modindex
- search
Python Module Index

e
  epitopepredict, 58
  epitopepredict.analysis, 35
  epitopepredict.app, 37
  epitopepredict.base, 38
  epitopepredict.config, 44
  epitopepredict.neo, 45
  epitopepredict.peptutils, 47
  epitopepredict.plotting, 48
  epitopepredict.sequtils, 49
  epitopepredict.tepitope, 52
  epitopepredict.tests, 53
  epitopepredict.tornado_serve, 54
  epitopepredict.utilities, 56
  epitopepredict.web, 57
add_dicts() (in module epitopepredict.utilities), 56
add_links() (epitopepredict.tornado_serve.NeoEpitopeHandler method), 55
add_path() (in module epitopepredict.app), 37
align_blast_results() (in module epitopepredict.analysis), 35
alignment_to_dataframe() (in module epitopepredict.analysis), 35
alignment_to_dataframe() (in module epitopepredict.sequtils), 49
allele_mapping() (epitopepredict.base.NetMHCIIPanPredictor method), 40
allele_summary() (epitopepredict.base.Predictor method), 41
allelenumber() (in module epitopepredict.tepitope), 52
analysis() (epitopepredict.app.WorkFlow method), 37
anchor_mutated() (in module epitopepredict.tepitope), 45

BasicMHCIPredictor (class in epitopepredict.base), 38
benchmark() (in module epitopepredict.tepitope), 52
binders_to_coords() (in module epitopepredict.dict.plotting), 48
blast_sequences() (in module epitopepredict.dict.sequitals), 49
bokeh_pie_chart() (in module epitopepredict.dict.plotting), 48
bokeh_plot_bar() (in module epitopepredict.dict.plotting), 48
bokeh_plot_grid() (in module epitopepredict.dict.plotting), 48
bokeh_plot_tracks() (in module epitopepredict.dict.plotting), 48
bokeh_summary_plot() (in module epitopepredict.plotting), 48
bokeh_test() (in module epitopepredict.plotting), 48
bokeh_vbar() (in module epitopepredict.plotting), 48

check_alleles() (epitopepredict.base.Predictor method), 41
check_alleles() (epitopepredict.base.TEpitopePredictor method), 43
check_enssembl() (in module epitopepredict.neo), 45
check_iedb_method() (in module epitopepredict.app), 37
check_iedbmhc1_path() (in module epitopepredict.app), 38
check_iedbmhc2_path() (in module epitopepredict.app), 38
check_imports() (in module epitopepredict.neo), 45
check_install() (epitopepredict.base.BasicMHCIPredictor method), 38
check_install() (epitopepredict.base.IEDBMHCIIPredictor method), 39
check_install() (epitopepredict.base.IEDBMHCIIPredictor method), 39
check_install() (epitopepredict.base.MHCFlurryPredictor method), 39
check_install() (epitopepredict.base.NetMHCIIPanPredictor method), 40
check_install() (epitopepredict.base.NetMHCPanPredictor method), 40
check_install() (epitopepredict.base.Predictor method), 41
epitopepredict Documentation

check_installed() (in module epitopepredict.app), 38
check_mhcl_length() (in module epitopepredict.app), 38
check_options() (in module epitopepredict.config), 44
check_snap() (in module epitopepredict.base), 43
check_tags() (in module epitopepredict.sequtils), 50
clean_sequence() (in module epitopepredict.base), 43
cleanup() (epitopepredict.base.Predictor method), 41
clustal_alignment() (in module epitopepredict.sequtils), 50
cm (epitopepredict.tornado_serve.ControlsForm attribute), 55
column_to_url() (in module epitopepredict.web), 57
combine() (epitopepredict.app.WorkFlow method), 37
combine_samples() (epitopepredict.neo.NeoEpitopeWorkFlow method), 45
combine_wt_scores() (in module epitopepredict.neo), 45
compare() (in module epitopepredict.tepitope), 52
compare_alleles() (in module epitopepredict.tepitope), 52
compare_anchor_positions() (in module epitopepredict.peptutils), 47
compare_predictors() (in module epitopepredict.base), 43
compare_ref() (in module epitopepredict.tepitope), 52
cmpare_tepitope_alleles() (in module epitopepredict.tepitope), 52
compress() (in module epitopepredict.utils), 56
ConfigForm (class in epitopepredict.tornadoServe), 54
ControlsForm (class in epitopepredict.tornadoServe), 55
convert_allele_name() (epitopepredict.base.MHCFlurryPredictor method), 39
convert_allele_name() (epitopepredict.base.NetMHCIIPanPredictor method), 40
convert_allele_name() (epitopepredict.base.NetMHCPanPredictor method), 40
convert_allele_names() (in module epitopepredict.tepitope), 52
convert_sequence_format() (in module epitopepredict.sequtils), 50
copyfile() (in module epitopepredict.utils), 56
copyfiles() (in module epitopepredict.utils), 56
cpus (epitopepredict.tornadoServe.ControlsForm attribute), 54
create_bokeh_table() (in module epitopepredict.web), 57
create_config_parser_from_dict() (in module epitopepredict.config), 45
create_figures() (in module epitopepredict.web), 57
create_frames() (in module epitopepredict.peptutils), 47
create_nmers() (in module epitopepredict.analysis), 35
create_random_peptides() (in module epitopepredict.peptutils), 47
create_random_sequences() (in module epitopepredict.peptutils), 48
create_sequence_html() (in module epitopepredict.sequence), 57
create_virtual_pssm() (in module epitopepredict.tepitope), 52
create_widgets() (in module epitopepredict.web), 57
cutoff (epitopepredict.tornadoServe.ControlsForm attribute), 55
cutoff_method (epitopepredict.tornadoServe.ControlsForm attribute), 55
dataframe_to_fasta() (in module epitopepredict.sequtils), 50
dataframe_to_seqrecords() (in module epitopepredict.sequtils), 50
dataframe_to_vcf() (in module epitopepredict.neo), 45
DataFrameIterator (class in epitopepredict.base), 38
dataframes_to_html() (in module epitopepredict.web), 57
dbscan() (in module epitopepredict.analysis), 35
deletecached (epitopepredict.tornadoServe.SummaryForm attribute), 56
dict_to_html() (in module epitopepredict.sequence), 56
dict_to_html() (in module epitopepredict.sequence), 56
distance_tree() (in module epitopepredict.web), 57
draw_genome_map() (in module epitopepredict.sequtils), 55
draw_labels() (in module epitopepredict.plotting), 48
DummyPredictor (class in epitopepredict.base), 39

E

effects_to_dataframe() (in module epitopepredict.neo), 45
effects_to_pickle() (in module epitopepredict.neo), 45
eml_to_dataframe() (in module epitopepredict.sequtils), 50
epitope_conservation() (in module epitopepredict.analysis), 35
epitopepredict (module), 58
epitopepredict.analysis (module), 35
epitopepredict.app (module), 37
epitopepredict.base (module), 38
epitopepredict.config (module), 44
epitopepredict.neo (module), 45
epitopepredict.peptutils (module), 47
epitopepredict.plotting (module), 48
epitopepredict.sequtils (module), 49
epitopepredict.tepitope (module), 52
epitopepredict.tests (module), 53
epitopepredict.tornado_serve (module), 54
epitopepredict.utilities (module), 56
epitopepredict.web (module), 57
ete_tree() (in module epitopepredict.sequtils), 50
evaluate() (epitopepredict.base.Predictor method), 41
exists() (in module epitopepredict.Predictor), 50
find_keyword() (in module epitopepredict.sequtils), 50
find_matches() (in module epitopepredict.neo), 46
first() (in module epitopepredict.base), 43
format_alignment() (in module epitopepredict.sequtils), 50
format_row() (epitopepredict.base.Predictor method), 41

G
genbank_summary() (in module epitopepredict.sequtils), 50
genbank_to_dataframe() (in module epitopepredict.sequtils), 50
generate_pssm() (in module epitopepredict.tepitope), 52
get() (epitopepredict.tornado_serve.DownloadHandler method), 55
get() (epitopepredict.tornado_serve.GlobalViewHandler method), 55
get() (epitopepredict.tornado_serve.MainHandler method), 55
get() (epitopepredict.tornado_serve.MakeConfigHandler method), 55
get() (epitopepredict.tornado_serve.NeoEpitopeHandler method), 55
get() (epitopepredict.tornado_serve.SequenceViewHandler method), 56
get_AAcontent() (in module epitopepredict.analysis), 36
get_AAfraction() (in module epitopepredict.peptutils), 48
get_AAsubstitutions() (in module epitopepredict.peptutils), 48
get_all_fragments() (in module epitopepredict.peptutils), 48
get_allele_cutoffs() (epitopepredict.base.Predictor method), 41
get_allele_data() (epitopepredict.base.IEDBMHCIPredictor method), 39
get_allele_pocket_sequences() (in module epitopepredict.tepitope), 52
get_alleles() (epitopepredict.base.BasicMHCIPredictor method), 38
get_alleles() (epitopepredict.base.IEDBMHCIPredictor method), 39
get_alleles() (epitopepredict.base.IEDBMHCIPredictor method), 39
get_alleles() (epitopepredict.base.MHCFlurryPredictor method),

Index 65
epitopepredict Documentation

get_alleles() (epitopepredict.base.MHCNuggetsPredictor method),
get_alleles() (epitopepredict.base.NetMHCIIIPanPredictor method),
get_alleles() (epitopepredict.base.NetMHCPanPredictor method),
get_alleles() (epitopepredict.base.TEpitopePredictor method),
get_alleles() (in module epitopepredict.app),
get_alleles() (in module epitopepredict.neo),
get_alleles() (in module epitopepredict.tepitope),
get_alleles() (in module epitopepredict.web),
get_args() (in module epitopepredict.tornado_server),
get_binders() (epitopepredict.base.Predictor method),
get_blast_results() (in module epitopepredict.sequils),
get_bokeh_colors() (in module epitopepredict.plotting),
get_cds() (in module epitopepredict.sequils),
get_closest_match() (in module epitopepredict.neo),
get_closest_matches() (in module epitopepredict.neo),
get.coords() (in module epitopepredict.base),
get_filenames() (in module epitopepredict.web),
get_features_qualifier() (in module epitopepredict.sequils),
get_file_labels() (epitopepredict.neo.NeoEpitopeWorkFlow method),
get_file_lists() (in module epitopepredict.web),
get_filenames() (in module epitopepredict.web),
get_fragments() (in module epitopepredict.peptutils),
get_genes_by_location() (in module epitopepredict.sequils),
get_global_rank() (epitopepredict.base.Predictor method),
get_identity() (in module epitopepredict.sequils),
get_iedb_request() (in module epitopepredict.base),
get_length() (in module epitopepredict.base),
get_matrix() (in module epitopepredict.tepitope),
get_mutant_sequences() (in module epitopepredict.neo),
get_names() (epitopepredict.base.Predictor method),
get_nearest() (in module epitopepredict.base),
get_options() (in module epitopepredict.config),
get_orthologs() (in module epitopepredict.analysis),
get_overlapping() (in module epitopepredict.base),
get_overlaps() (in module epitopepredict.analysis),
get_pocket_positions() (in module epitopepredict.tepitope),
get_pockets_pseudo_sequence() (in module epitopepredict.tepitope),
get_pos() (in module epitopepredict.base),
get_predictor() (in module epitopepredict.base),
get_predictor_classes() (in module epitopepredict.base),
get_predictors() (in module epitopepredict.web),
get_preset_alleles() (in module epitopepredict.base),
get_pseudo_sequence() (in module epitopepredict.tepitope),
get_pssm_score() (in module epitopepredict.tepitope),
get_pssms() (in module epitopepredict.tepitope),
get_quantile_data() (epitopepredict.base.Predictor method),
get_quantiles() (in module epitopepredict.base),
get_ranking() (epitopepredict.base.Predictor method),
get_results_info() (in module epitopepredict.web),
get_results_tables() (in module epitopepredict.web),
get_scores() (epitopepredict.base.Predictor method),
get_scores() (in module epitopepredict.tepitope),
get_seq_from_binders() (in module epitopepredict.plotting),
get_seqdepot() (in module epitopepredict.analysis),
get_seqdepot_annotation() (in module epitopepredict.plotting),
get_sequence() (in module epitopepredict.base), 44
get_sequence() (in module epitopepredict.sequtils), 51
get_sequencefrom_pdb() (in module epitopepredict.base), 57
get_sequences() (in module epitopepredict.app), 38
get_sequences() (in module epitopepredict.web), 58
get_similarities() (in module epitopepredict.tepitope), 52
get_species_name() (in module epitopepredict.analysis), 36
get_standard_mhc1() (in module epitopepredict.base), 44
get_standard_mhc2() (in module epitopepredict.base), 44
get_summary() (epitopepredict.app.WorkFlow method), 37
get_summary_tables() (in module epitopepredict.web), 58
get_translation() (in module epitopepredict.sequtils), 51
get_unique_cores() (epitopepredict.base.Predictor method), 41
get_url() (epitopepredict.tornado_serve.SequenceViewHandler method), 56
get_variant_class() (in module epitopepredict.neo), 46
get_variants_effects() (in module epitopepredict.neo), 46
GlobalViewHandler (class in epitopepredict.tornado_serve), 55

H
help_msg() (in module epitopepredict.tornado_server), 56

I
i (epitopepredict.tornado_server.ConfigForm attribute), 54
i (epitopepredict.tornado_server.ControlsForm attribute), 55
iedb_checks() (in module epitopepredict.app), 56
iedbmhc1_path (epitopepredict.tornado_server.ConfigForm attribute), 54
iedbmhc2_path (epitopepredict.tornado_server.ConfigForm attribute), 54

I
IEDBMHCIIPredictor (class in epitopepredict.base), 39
IEDBMHCIIPredictor (class in epitopepredict.base), 39
index_genbank_features() (in module epitopepredict.sequtils), 51
is_seqfile() (in module epitopepredict.tornado_server), 56
isoelectric_point() (in module epitopepredict.analysis), 36

K
kind (epitopepredict.tornado_server.ControlsForm attribute), 55
kinds (epitopepredict.tornado_server.ControlsForm attribute), 55

L
list_alleles() (in module epitopepredict.app), 38
load() (epitopepredict.base.Predictor method), 41
load_variants() (in module epitopepredict.neo), 46
local_blast() (in module epitopepredict.sequtils), 51

M
main() (in module epitopepredict.app), 38
main() (in module epitopepredict.peptutils), 48
main() (in module epitopepredict.tepitope), 53
MainHandler (class in epitopepredict.tornado_server), 55
make_blastdb() (in module epitopepredict.neo), 46
make_humanblastdb() (in module epitopepredict.neo), 46
make_virusblastdb() (in module epitopepredict.neo), 46
MakeConfigHandler (class in epitopepredict.tornado_server), 55
mhcl_alleles (epitopepredict.tornado_server.ConfigForm attribute), 54
mhcl_length (epitopepredict.tornado_server.ConfigForm attribute), 54
mhcl_presets (epitopepredict.tornado_server.ConfigForm attribute), 54
mhcl2_alleles (epitopepredict.tornado_server.ConfigForm attribute), 54
mhcl2_length (epitopepredict.tornado_server.ConfigForm attribute), 54
mhc2_presets (epitopepredict.tornado_serve.ConfigForm attribute), 54
MHCFlurryPredictor (class in epitopepredict.base), 39
MHCNuggetsPredictor (class in epitopepredict.base), 40
muscle_alignment() (in module epitopepredict.sequtils), 51

N
n (epitopepredict.tornado_serve.ControlsForm attribute), 55
name (epitopepredict.tornado_serve.ControlsForm attribute), 55
needle_alignment() (in module epitopepredict.sequtils), 51
NeoEpitopeHandler (class in epitopepredict.tornado_serve), 55
NeoEpitopeWorkFlow (class in epitopepredict.neo), 45
NeoForm (class in epitopepredict.tornado_serve), 56
net_charge() (in module epitopepredict.analysis), 36
net_charge() (in module epitopepredict.peptutils), 48
NetMHCIIIPanPredictor (class in epitopepredict.base), 40
NetMHCPanPredictor (class in epitopepredict.base), 40
next() (epitopepredict.base.DataFrameIterator method), 39
no_data_render() (epitopepredict.tornado_serve.NeoEpitopeHandler method), 55

O
overwrite (epitopepredict.tornado_serve.ConfigForm attribute), 54

P
p1 (epitopepredict.tornado_serve.ConfigForm attribute), 54
p2 (epitopepredict.tornado_serve.ConfigForm attribute), 54
pairwise_alignment() (in module epitopepredict.sequtils), 51
parse_config() (in module epitopepredict.config), 45
path (epitopepredict.tornado_serve.ConfigForm attribute), 54
pmec_score() (in module epitopepredict.neo), 46
peptide_properties() (in module epitopepredict.analysis), 37
peptides_from_effect() (in module epitopepredict.neo), 46
pickpocket() (in module epitopepredict.tepitope), 53
plot() (epitopepredict.base.Predictor method), 41
plot_bars() (in module epitopepredict.plotting), 49
plot_bcell() (in module epitopepredict.plotting), 49
plot_binder_map() (in module epitopepredict.plotting), 49
plot_heatmap() (in module epitopepredict.plotting), 49
plot_multiple() (in module epitopepredict.plotting), 49
plot_overview() (in module epitopepredict.plotting), 49
plot_regions() (in module epitopepredict.plotting), 49
plot_results() (epitopepredict.app.WorkFlow method), 37
plot_seqdepot() (in module epitopepredict.plotting), 49
plot_summary_heatmap() (in module epitopepredict.base), 44
plot_tracks() (in module epitopepredict.plotting), 49
plot_variant_summary() (in module epitopepredict.neo), 46

pm (epitopepredict.tornado_serve.ConfigForm attribute), 54
predict() (epitopepredict.base.BasicMHCIPredictor method), 38
predict() (epitopepredict.base.DummyPredictor method), 39
predict() (epitopepredict.base.IEDBMHCIIIPredictor method), 39
predict() (epitopepredict.base.IEDBMHCIPredictor method), 39
predict() (epitopepredict.base.MHCFlurryPredictor method), 39
predict() (epitopepredict.base.MHCNuggetsPredictor method), 40
predict() (epitopepredict.base.NetMHCIIIPanPredictor method), 40
predict() (epitopepredict.base.NetMHCPanPredictor method), 40
predict() (epitopepredict.base.TEpitopePredictor method), 42
predict_binding() (in module epitopepredict.neo), 46
predict_peptides() (epitopepredict.base.Predictor method), 42
predict_peptides_worker() (in module epitopepredict.neo), 46

Index
score_peptides()  (in module epitopepredict.neo), 47
search_pubmed()  (in module epitopepredict.utilities), 57
self_matches()  (in module epitopepredict.neo), 47
self_similarity()  (in module epitopepredict.neo), 47
seq_from_binders()  (in module epitopepredict.base), 44
seqdepot_to_coords()  (in module epitopepredict.plotting), 49
seqs_to_dataframe()  (in module epitopepredict.sequtils), 51
sequence_file()  (epitopepredict.base.Predictor method), 42
sequence_from_peptides()  (in module epitopepredict.web), 58
sequence_to_html_grid()  (in module epitopepredict.web), 58
sequences_to_html_table()  (in module epitopepredict.web), 58
SequenceViewHandler()  (class in epitopepredict.tornado_serve), 56
set_defaults()  (in module epitopepredict.app), 38
set_netmhcpan_cmd()  (in module epitopepredict.base), 44
setUp()  (epitopepredict.app.WorkFlow method), 37
setUp()  (epitopepredict.neo.NeoEpitopeWorkFlow method), 45
setUp()  (epitopepredict.tests.PredictorTests method), 53
show_alignment()  (in module epitopepredict.sequtils), 51
show_alignment_html()  (in module epitopepredict.sequtils), 51
show_pocket_residues()  (in module epitopepredict.tepitope), 53
show_predictors()  (in module epitopepredict.app), 38
show_predictors()  (in module epitopepredict.neo), 47
show_preset_alleles()  (in module epitopepredict.app), 38
signalP()  (in module epitopepredict.analysis), 37
similarity_score()  (in module epitopepredict.tepitope), 53
split_peptides()  (in module epitopepredict.base), 44
str_to_html()  (in module epitopepredict.tornado_serve), 56
summarize()  (epitopepredict.base.Predictor method), 43
summarize()  (in module epitopepredict.base), 44
summarize_by_protein()  (in module epitopepredict.base), 44
summary_plots()  (in module epitopepredict.neo), 47
SummaryForm()  (class in epitopepredict.tornado_serve), 56
supported_lengths()  (epitopepredict.base.BasicMHCIPredictor method), 38
supported_lengths()  (epitopepredict.base.Predictor method), 43
supported_lengths()  (epitopepredict.base.TEpitopePredictor method), 43
symmetrize()  (in module epitopepredict.utilities), 57
T
tabbed_html()  (in module epitopepredict.web), 58
TEpitopePredictor()  (class in epitopepredict.base), 43
test()  (in module epitopepredict.analysis), 37
test()  (in module epitopepredict.tepitope), 53
test()  (in module epitopepredict.utilities), 57
test()  (in module epitopepredict.web), 58
test_basicmhc1()  (epitopepredict.tests.PredictorTests method), 53
test_binary()  (in module epitopepredict.app), 38
test_classes()  (epitopepredict.tests.PredictorTests method), 53
test_conservation()  (in module epitopepredict.analysis), 37
test_cutoffs()  (epitopepredict.tests.PredictorTests method), 53
test_fasta()  (epitopepredict.tests.PredictorTests method), 53
test_features()  (epitopepredict.tests.PredictorTests method), 53
test_features()  (in module epitopepredict.analysis), 37
test_iedbmhc1()  (epitopepredict.tests.PredictorTests method), 53
test_load()  (epitopepredict.tests.PredictorTests method), 53
test_mhcflurry()  (epitopepredict.tests.PredictorTests method), 54
test_mhcflurry()  (epitopepredict.tests.PredictorTests method), 54
test_mhcflurry()  (epitopepredict.tests.PredictorTests method), 54
test_multiproc()  (epitopepredict.tests.PredictorTests method), 54
test_netmhcpan()  (epitopepredict.tests.PredictorTests method), 54
test_peptide_prediction()  (epitopepredict.tests.PredictorTests method), 54
test_run()  (in module epitopepredict.app), 38
test_run()  (in module epitopepredict.neo), 47
test_tepitope() (epitopepredict.dict.tests.PredictorTests method), 54

testrun() (in module epitopepredict.analysis), 37

tmhmm() (in module epitopepredict.analysis), 37

V

varcode_test() (in module epitopepredict.neo), 47

variants_from_csv() (in module epitopepredict.neo), 47

venndiagram() (in module epitopepredict.utilities), 57

view (epitopepredict.tornado_serve.ControlsForm attribute), 55

view (epitopepredict.tornado_serve.NeoForm attribute), 56

views (epitopepredict.tornado_serve.ControlsForm attribute), 55

views (epitopepredict.tornado_serve.NeoForm attribute), 56

virus_matches() (in module epitopepredict.neo), 47

virus_similarity() (in module epitopepredict.neo), 47

W

WorkFlow (class in epitopepredict.app), 37

write_config() (in module epitopepredict.config), 45

write_default_config() (in module epitopepredict.config), 45

write_fasta() (in module epitopepredict.base), 44

write_seqs() (epitopepredict.base.MHCNuggetsPredictor method), 40

wt_similarity() (in module epitopepredict.neo), 47

X

x (epitopepredict.tornado_serve.ConfigForm attribute), 54