epitopepredict: A tool for integrated MHC binding prediction

Damien Farrell[1¶]

[1]UCD School of Veterinary Medicine, University College Dublin, Ireland.

[¶]Corresponding Author

Dr. Damien Farrell, E-mail: farrell.damien@gmail.com

Keywords:

MHC-binding; immunoinformatics; antigen; epitope; T-cell

Short title: epitopepredict

Abstract

A key step in the cellular adaptive immune response is the presentation of antigen to T cells. During this process short peptides processed from self or foreign proteins may be presented on the surface bound to MHC molecules for binding to T cell receptors. Those that bind and activate an immune response are called epitopes. Computational prediction of T cell epitopes has many applications in vaccine design and immuno-diagnostics. This is the basis of immunoinformatics which allows in silico screening of peptides before experiments are performed. The most effective approach is to estimate the binding affinity of a given peptide fragment to MHC class I or II molecules. With the availability of whole genomes for many microbial species it is now feasible to computationally screen whole proteomes for candidate peptides. epitopepredict is a programmatic framework and command line tool designed to aid this process. It provides access to multiple binding prediction algorithms under a single interface and scales for whole genomes using multiple target MHC alleles. A web interface is provided to assist visualization and filtering of the results. The software is freely available under an open source license from https://github.com/dmnfarrell/epitopepredict

Background

An essential step in provoking adaptive immunity, delivered by the activated CD8+ or CD4+ T cells, is the recognition of T cell receptor (TCR) to T cell epitopes. The 'epitope' is the peptide-MHC combination resulting from the binding of antigenic peptides to MHC proteins. This is the major determinant step and is computationally predictable. Algorithms that can identify MHC-class I ot MHC-class II binding peptides rapidly and accurately are essential in for vaccine development, neo-epitope discovery and immunogenicity screening of protein therapeutics. Many MHC binding predic-

tion methods exist for both class I and II and have been comprehensively reviewed [1]. Currently the most effective are machine learning (ML) based approaches which are trained on existing binding affinity data for a given MHC molecule. To do this the peptide sequence is encoded and these features fit against the known affinity. To date artificial neural networks (ANN) preform better at this task than other models such as linear regression. This is likely because the hidden layers in such networks are better able to account for the contribution of intrapeptide residue-residue interactions to the binding affinity. All methods vary in accuracy over MHC alleles depending on the availability of quality datasets. Pan-allele tools have been developed to deal with this issue [2]. These approaches can impute affinities for unknown alleles on the basis of neighboring MHC with the highest sequence similarity and which have sufficient training data.

By convention, selection of peptides is done using an arbitrary score threshold. For affinities, a threshold value of 500nM is considered a binder and 50nM a strong binder. The algorithms perform best at this classification task rather than re-producing exact affinities. This problem is intrinsic to ML-based approaches: the effect of the most dominant features is penalized intentionally to achieve better generalization on blind test data [3]. Another source of the inaccuracy is the loss of sensitivity of experimental assays at either very high or low binding affinity regimes. As a consequence, epitope candidates for subsequent experimental validation selected by ranking the affinities may not necessarily be the best approach. Percentage ranking is now often the recommended method [4]. However the exact approach probably depends on the study in question. For example, searching a small number of proteins might mean taking the top ranked percentile from each sequence regardless of score. Threshold selection is discussed later in the examples.

Strategies for epitope selection

A typical approach to binder selection is to select the top n^{th} percentile per protein rather than using an absolute threshold value; however for whole proteome studies this is likely to introduce multiple false positives from peptides in proteins that would otherwise score very low globally. We therefore include in our method a global standardization of the score over the entire proteome, similar to that used by Bremel et al. [5] and others, by setting a global cut-off based on the top percentage of scores from the entire proteome. In addition, some alleles have a significantly higher score distribution and will dominate the results if a uniform score cut-off is applied; this applies in general to MHC binding predictors. Thus separate global cut-off per allele so that low scoring alleles would be better represented is also advisable. This approach is consistent with recent work by Paul et al. [6] regarding allele-specific thresholds in MHC-I prediction. Three alternative such threshold strategies are provided in this library and discussed below.

Binding promiscuity

Promiscuous MHC binders are defined in this context as those above the cutoffs in more than a given number of 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.

Tools for epitope selection

Software for T cell vaccine development or neoepitope prediction currently concentrates on using the binding prediction or eluted ligand likelihood as the main selection methods. Typically when a binding prediction tool is published, the authors will provide a binary that can be used on the command line or via a web interface. Some tools provide both. Command line tools offer better control and perhaps higher throughput but may be harder to use for a general user. Virtually all of these require users separately input each sequence and it's allele. It is then difficult or impossible to integrate results from multiple sequences and alleles. The results are often in different formats and it is not possible to compare between algorithms, for example.

There are several computational pipelines that help a researcher to do epitope prediction [7,8]. Other commercial desktop software applications for epitope discovery are EpiMatrix [9]. Commercial tools may be of high quality but are neither free nor open source, raising issues of reproducibility for academics. Therefore there is a limited choice for users in readily available and easy to use tools.

Implementation

This software is implemented entirely in Python. To achieve some level of uniformity between prediction methods a standardized programmatic interface for executing the binding prediction methods and processing the results was designed. The results from each method can then be processed and visualized in a consistent manner. Prediction methods are implemented by inheriting from a Predictor object. Each predictor may wrap methods from other python packages or call command line predictors. For example the *TepitopePredictor* uses the epitopepredict.tepitope module provided with this package. This approach allows us to integrate a new prediction method in a relatively straightforward and consistent manner. The prediction methods always return a Pandas DataFrame [10] in a standard format. The *predict_sequences* method is used for multiple protein sequences and can be run in parallel. This can take a GenBank or fasta file as input. For large numbers of sequences the prediction function 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. More details on how to use the Python API are given in the online documentation and in the example notebooks referencing the examples below.

The web application is implemented in Tornado [11] using the Bokeh [12] visualization library for

making interactive plots.

Supported MHC binding prediction tools

The following MHC binding prediction methods are supported though the API. This means they can be utilized via the command line tool. The first two are built-in to the package, the others require installation of external software by the user. NetMHC tools in particular have to be installed separately as they have a more restrictive academic license that does not allow them to be distributed by a third party or via a repository. Only the 'pan specific' versions of these tools are supported as they provide the best allelic coverage.

- TEPITOPEpan [13] is a position specific scoring matrix (PSSM) based algorithm. It uses 11 scoring matrices derived from combinatorial competitive binding assays on 11 HLA-DR alleles [14]. This method is pan specific and covers 700 HLA-DR molecules with unknown binding specificities based on pocket similarity to the original set of 11 library sequences. We have implemented this algorithm as a Python module, thus it comes with the package. It is fast but not as accurate in benchmarks as netMHCIIpan with less alleles covered.
- The BasicMHC1 predictor is a built-in method MHC-I prediction method further detailed below. It is implemented using the scikit-learn [15] package. It only covers 103 MHC-I alleles and cannot currently be extrapolated to use with similar alleles (i.e. not pan specific) but provides a convenient alternative to the external tools.
- MHCflurry [16] is an MHC-I predictor also using ANNs trained on affinity measurements. It currently covers 112 human alleles. This is an open source tool available via pip and thus easy to install. It is recommended for MHC-I predictions unless there are alleles not covered. The latest supported version is 2.0.1.
- NetMHCpan [17] is an artificial neural network algorithm covering many human and animal MHC-I alleles. This is trained on both MS eluted ligand data and binding affinity data. It therefore returns two properties: either the likelihood of a peptide becoming a natural ligand, or the predicted binding affinity. Version 4.1 is currently supported.
- NetMHCIIpan [18] is also an ANN, trained on binding data for multiple MHC-II alleles. Predictions are now extended to all HLA-DR, DQ and DP known sequences as from version 3.0 [19]. Both this tool and netMHCpan have the broadest species support of any algorithms. They both have good web interfaces but are covered by free non-commercial academic licenses and the local versions must be installed separately. Version 3.0 is supported.

Available threshold methods

Thresholds for considering a peptide to be a binder are somewhat arbitrary. This tool provides three threshold methods. The results from each will overlap but will not be identical. These are applied per sequence/protein and per each allele using the currently loaded data. These three threshold methods are also available when calculating promiscuous binders. Ultimately these are simply alternative methods of achieving the same result - reducing the set of predicted peptides.

rank – Selects the top ranking peptides in each sequence above a rank cutoff. This is the most frequently recommended method of binder selection in general.

score - Uses a single score cutoff for all peptides. Most binding predictors produce a binding affinity score (ic50) and a cutoff of 500nM is common. There is no rule over which score cutoff is optimal however. Some alleles will tend to produce higher scores. Also unless some limit is placed on the number of peptides, large proteins will produce a lot of peptides compared to smaller sequences.

global - Allele specific 'global' cutoffs, this uses a percentile cutoff to select peptides using pre-calculated quantile scores for each allele. The global quantile scores were calculated for each prediction method using a set of sequences from known human antigens such as apical membrane antigen, Tetanus toxin, thrombopoietin and interferon beta. Therefore peptides can be selected as measured against a standard scale as opposed to their 'within protein' ranking. A typical value would be using the top 5% in each allele across all sequences. This technique is designed for selection of a small set of candidates from very large numbers of proteins such as across a bacterial proteome. There is limited evidence to suggest this is superior to the other methods but we have used it for selection of a small set of candidates from large numbers of proteins, detailed in example 2 below.

A basic MHC-I predictor

This section details the built-in method for MHC-I binding prediction. It is implemented in Python using scikit-learn. The typical method of building such an algorithm is to encode the peptide amino acid sequences numerically in a manner that captures the properties important for binding. Then these features can be fit against their known binding affinities (or eluted ligand data) using a regression model of some kind. Several peptide encoding schemes were tested, including the NLF encoding scheme [20], OETMAP [21], a Blosum62 matrix or a simple 'one hot' encoding method. One hot encoding was found to be adequate and the more complex schemes did not appear to offer any significant advantage. This may require further testing. For now it is possible to instantiate and train the predictor with any of these encoders. The regression model used is the *MLPRegressor* from sklearn, an implementation of a multilayer perceptron (MLP), a class of artificial neural network. The data used for training was primarily from the IEDB and was curated by

the authors of MHCflurry [16] from various sources. The regression model must be trained for each allele. When this is done the model is persisted with the joblib module and can be re-loaded for new predictions for that allele. All of this functionality is encapsulated in the *BasicMHCIPredictor* class in epitopepredict. The predictor only supports 103 alleles currently and is not pan specific as of yet. This feature is still to be added.

To test performance, a separate evaluation set of peptides originally created by Kim et al. [22] was downloaded from the IEDB. The training set sequences were subtracted from this leaving 25948 9-mer peptides. Only alleles for which there were more than 200 peptides were evaluated to give a reasonable performance estimate. This left 40 HLA alleles for testing. Both the Pearson correlation coefficient and the roc auc metric (with a threshold of below 500nM set as a positive binder) were used as metrics. The results in Figure 1 show that our predictor gives similar performance to the others with this test set. It is not meant to provide a definitive benchmark since these other tools have been more comprehensively benchmarked elsewhere. In particular it can be hard to obtain a benchmark set of peptides that has not been used for training in one or more of the models.

In practical use this predictor can be run directly from the API or command line without installing any other program. Models are trained once as needed for each allele/length combination using the current installed versions of scikit-learn and joblib. Once trained each model is saved and can be re-used. Training only takes a matter of seconds for each model.

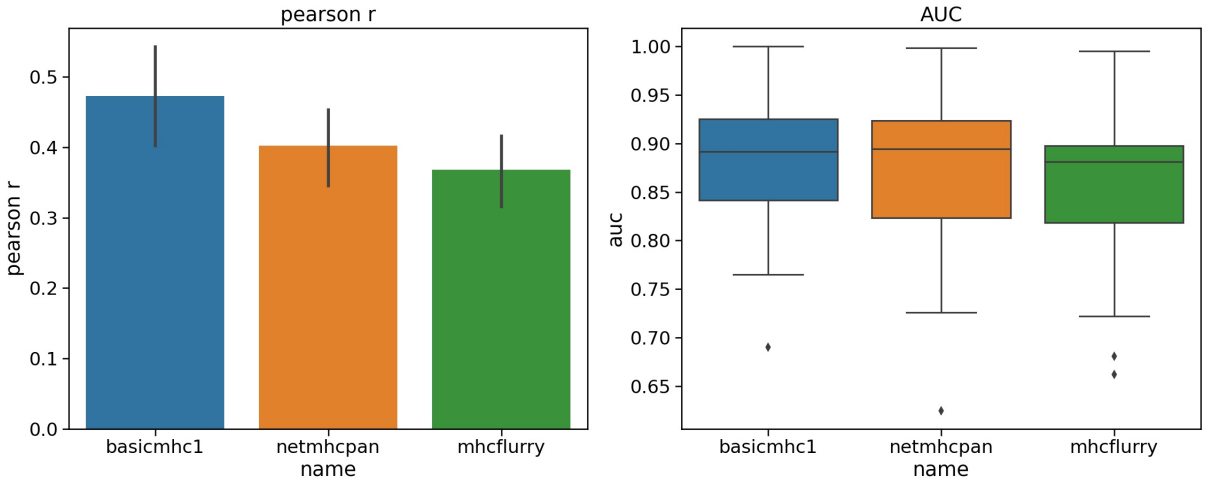

Figure 1: Performance of the basicmhc1 predictor compared to netMHCpan and MHCflurry for 40 human alleles. (a) Mean Pearson r and (b) mean AUC scores over all alleles. Only alleles with evaluation data for over more than 200 peptides were used. This test dataset used 9-mer peptides only.

Results

In the following we use several examples to illustrate the use of this package in practice with real data. These are available as Jupyter notebooks stored at https://github.com/dmnfarrell/epitopepredict/tree/master/examples. They are also archived permanently on Zenodo and the latest version can be found using the DOI: 10.5281/zenodo.593878. Some of these notebooks are also reproducible using the epitopepredict examples Code Ocean capsule (DOI: 10.24433/CO.5815986.v1).

Example 1: Predictions for selected antigens in Mycobacterium Tuberculosis – comparison with experimental data

A typical use of epitope prediction tools is to select a candidate list of peptides for testing from a large sequence space representing multiple potential antigens. This example provides a comparison of the three different selection methods in epitopepredict using a realistic example. It uses a set of known CD4 epitopes discovered in a study by measuring IFN-γ T cell responses to M. tuberculosis (Mtb) antigens in a healthy South African cohort [23]. The test data is available as supplementary tables in that paper. It comprises 75 15-mer epitopes selected from a set of known Mtb antigens. Here we perform a simple benchmark to find the percentage coverage of predicted MHC-II binders in two predictors, netMHCIIpan and Tepitope, using the three threshold methods for selecting promiscuous binders described above. These are then compared across a selection of cut-offs that each yield a certain number of binders. Ideally we would want to produce as small a number of predicted binders as possible to reduce the number to be experimentally tested.

The sequences of all 29 proteins represented in the target set where retrieved and split into 15-mers. Then predictions were made for each of the 27 alleles in the target population tested in the study. This produces a list of 9,299 peptides predicted for each allele. With epitopepredict selection of promiscuous binders can be done easily with a single command. Binders promiscuous above thresholds in at least five alleles were selected.

The results are shown in Figure 2, with the plots showing the percentage of experimental peptides covered versus the number of predicted binders, corresponding to a certain cut-off in each method. It is seen that the ‘rank’ method is superior in both cases as it achieves a higher coverage with the lowest number of binders. All the curves level off at about 80% coverage. The ‘rank’ method may work better in this case partly because some of the epitopes were originally selected by prediction algorithms using a similar approach.

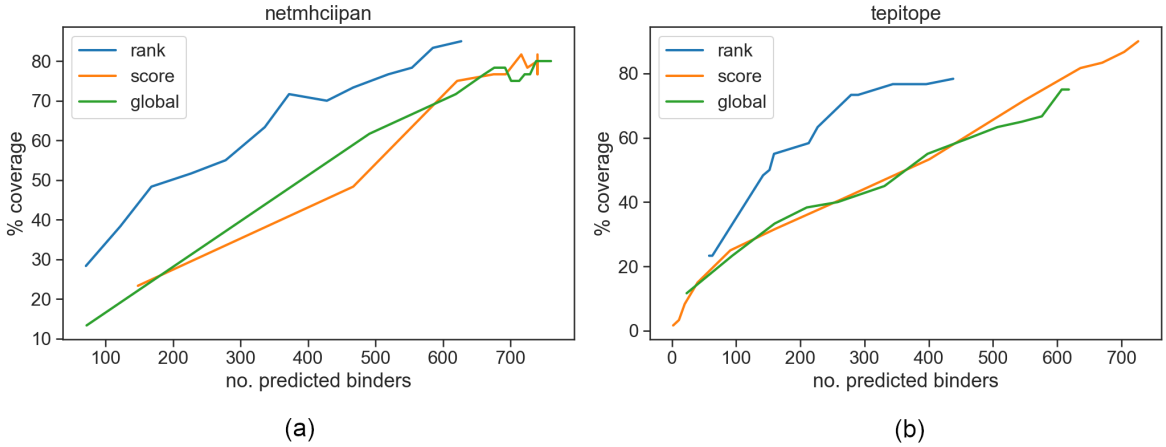

Figure 2: Performance of three binder selection methods showing the percentage coverage of experimental positive peptides by predicted binders at different cutoff levels. The higher the cutoff the more binders are predicted until the curves level off. Results are shown for (a) netMHCIIpan and (b) Tepitope.

Example 2: Scanning the proteome of Mycobacterium bovis for CD4+ epitopes

We have previously used this package to prioritize CD4+ epitopes in the proteome of M. bovis (*Mycobacterium tuberculosis* variant bovis AF2122/97) for potential use in novel antigens for bovine tuberculosis [24]. The results are documented in the paper. Briefly, we performed binding predictions over the entire M. *Bovis* proteome using two different binding predictors, netMHCIIpan [19], Tepitope [13]. For each set of results we found only promiscuous binders above an allele specific cutoff using the 'global' selection strategy. In addition clusters of binders were detected to find areas of high binder density in each sequence. The assumption underlying this method is that ~20mer peptides covering these regions will be more likely to yield at least one true positive epitope and hence elicit a T-cell response. The results are a set of clusters for both prediction methods, ranked by number of binders per unit length. This has also been referred to as the 'epitope density' method [25]. We further contrasted this cluster selection with the more conventional ranking of top scoring binders. We also included random non high scoring peptides as a control. 20-mer peptides derived from these sets were synthesized and tested for IFN-γ responses in M.bovis naturally infected cattle. Approximately 24% out of 270 peptides had high responses (using known epitopes as the baseline response). The random controls had no responses above this threshold.

This workflow was performed using an older version of this software. A newer and somewhat simplified form of the same analysis is now available as a notebook in the examples folder. Results from this output will be slightly different to our previous analysis since some of the extra steps have been removed but the methodology is the same.

Example 3: Predicting cross-reactive T cell epitopes in Sars-CoV-2

Eight months after the initial outbreak, puzzles remain about the human immune response to the SARS-CoV-2 virus. By now a significant proportion in some large cities, such as New York, will have been exposed. However antibody tests have often revealed lower than expected rates of seropositivity in populations where the virus has spread [26]. It is almost certain that other components of the immune system are important in protecting individuals just as in other infectious diseases. Robust innate immune responses are one candidate. Another possibility is T cells. SARS-CoV-2 reactive CD4+ T cells have been reported in unexposed individuals, suggesting pre-existing cross-reactive T cell memory in 20-50% of people [27]. It is possible these are memory T cells generated from previous exposures to the human common cold coronaviruses (HCoVs) which circulate widely.

Mateus et al. [28] have identified such cross-reactive CD4+ epitopes by generating 42 short term T cell lines specific to previously identified epitopes in PBMCs from unexposed donors. Then homologs to these peptides in the HCoVs were tested against these cell lines for a response. These tests were done in both unexposed and convalescent COVID19 patients. Cross reactivity was found in 10/42 of the T cell lines. Responding cells in unexposed donors were predominantly found in the effector memory CD4+ T cell population, though the consequences of this for protective immunity are not yet known.

Here we show how it's possible to predict such potential cross-reactive CD4+ epitopes just using the sequences.

The method used is as follows:

- Predict MHC-binders in each SCoV2 protein sequence and selected the top scoring candidates. Here we use epitopepredict to predict the most promiscuous binders across the 8 most representative human MHC-II alleles. Each protein sequence is split into 15-mer peptides and scored.
- Select the top scoring peptides in each protein. In this case we select the peptides using the global cutoff method in the top 5% percentile for each allele. We also limit the total for each protein to 70 to prevent a very long protein like ORF1ab from dominating the selection.
- Calculate conservation of each peptide with it's closest homologous sequence in each of the other four HCoVs. Then rank them by percentage identity.

Using a limit of 70 peptides per protein we found 282 predicted peptides. Out of these, 162 were conserved with >67% identity in at least one HCoV (most commonly with Sars). Note that for a peptide to be cross-reactive it does not necessarily have to share all residues in common with it's homolog. The 9-mer core binding sequence could be conserved with perhaps similar residues at

ends. We finally checked our 162 peptides against the 10 epitopes identified by Mateus et al. We found a hit in 6/10 cases, shown in Table 1. Some hits are two peptides in our set overlapping which probably indicates the same core epitope.

Sequence	Protein	Start	Hit from Predicted Set
PSGTWLTYTGAIKLD	N	326	GTWLTYTGAIKLDDK
SFIEDLLFNKVTLAD	S	816	FIEDLLFNKVTLADA, DLLFNKVTLADAGFI
YEQYIKWPWYIWLGF	S	1206	None
VLKKLKKSLNVAKSE	nsp8	3976	VVLKKLKKSLNVAKS, EVVLKKLKKSLNVAK
KLLKSIAATRGATVV	nsp12	4966	RQFHQKLLKSIAATR
EFYAYLRKHFSMMIL	nsp12	5136	NEFYAYLRKHFSMMI, YLRKHFSMMILSDDA
LMIERFVSLAIDAYP	nsp12	5246	None
TSHKLVLSVNPYVCN	nsp13	5361	None
NVNRFNVAITRAKVG	nsp13	5881	VNRFNVAITRAKVGI

Table 1: Matches to the 10 cross reactive peptides found by Mateus et al. from our predicted binders shows hits in 6/10 cases.

Usage

Command Line Interface

Installing the package provides a command line tool that is run from a terminal. It is envisaged that most users will utilize the package using this tool since it requires no programming knowledge. It provides pre-defined functionality with all inputs and settings specified in a text configuration file. One advantage of using configuration files is in avoiding long commands with multiple arguments that may be prone to causing errors. Also configuration files can be kept to recall what setting was used for a particular workflow. Using this you can make MHC predictions with your chosen alleles and predictors in one run. If settings are left out generally defaults will be used so one can use a minimal file, simplifying usage. Other useful features of the tool are the ability to run predictions in parallel using multiple processing cores, the use of preset lists of alleles and resuming runs that have been interrupted without overwriting previous predictions. Results are saved to disk as text files and can be re-read in a subsequent run of the tool without having to re-calculate binding predictions.

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.

API usage

A very basic example of how to use the library from the Python API is given here. More complex usage is detailed in the documentation.

```
import epitopepredict as ep
P = ep.get_predictor('basicmhc1')
from epitopepredict import peptutils
#get some random peptides, returns a list
seqs = peptutils.create_random_sequences(10)
#run predictions
res = P.predict_peptides(seqs, alleles='HLA-A*01:01')
```

The above code returns a pandas DataFrame sorted by allele and rank.

Plotting

The API includes the ability to plot results for individual protein sequences for one or more predictor. In such plots binders are shown as colored blocks at their position in the protein with multiple tracks, one per allele/method. This allows ready comparisons between methods. An example is shown in Figure 3. This shows binders for three MHC-class I predictors for an antigenic Mtb protein, Rv3875. Six HLA alleles are shown. We can see that each method has some overlap with the others.

Testing

The command line tool can be tested by calling `epitopepredict -t` which runs a set of sample Ebola virus sequences with the available prediction methods. Outputs are saved to a folder called zaire_test. It should be noted that this is not used as a benchmark test since the algorithms used have all been tested independently. This is an example run for the user to check that the command line workflow is working and to inspect the outputs.

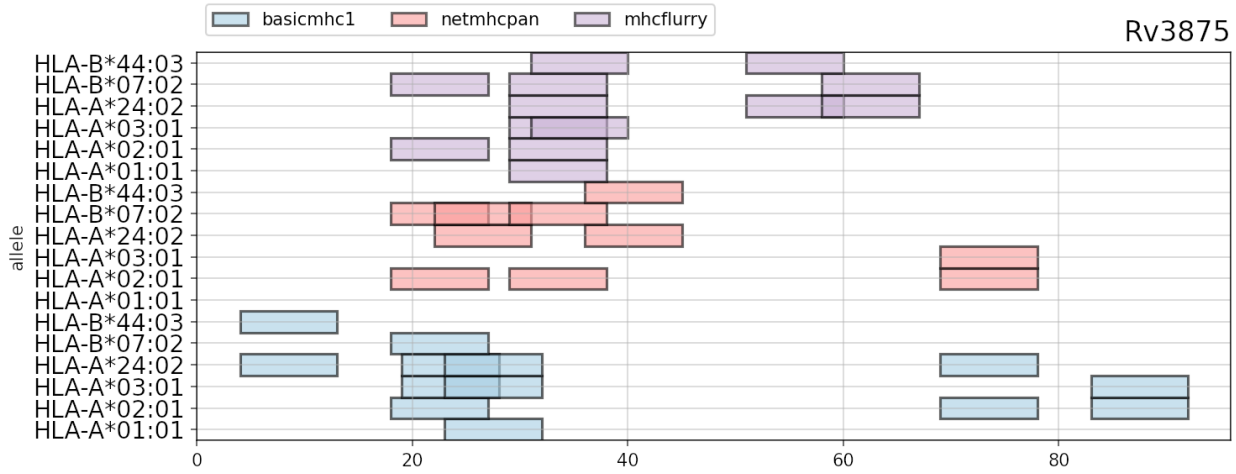

Figure 3: Predicted 'promiscuous' binders in a sample sequence for three methods. Each method will have some overlapping peptides but they are usually likely to differ.

Web Application

A web interface that is launched from the command line can be used to view results from a set of predictions that have been previously made. This is an improved and much easier to use form of a previous web interface called epitopemap [29] and replaces it. Widgets can be used to select thresholds and the kind of plot shown. Currently two kinds of plots can be viewed, a sequence view and one that shows the peptides as colored blocks in tracks along the sequence, as shown in Figure 4. This web interface can be tested by running the test command above and then launching the web app using the zaire_test folder as input.

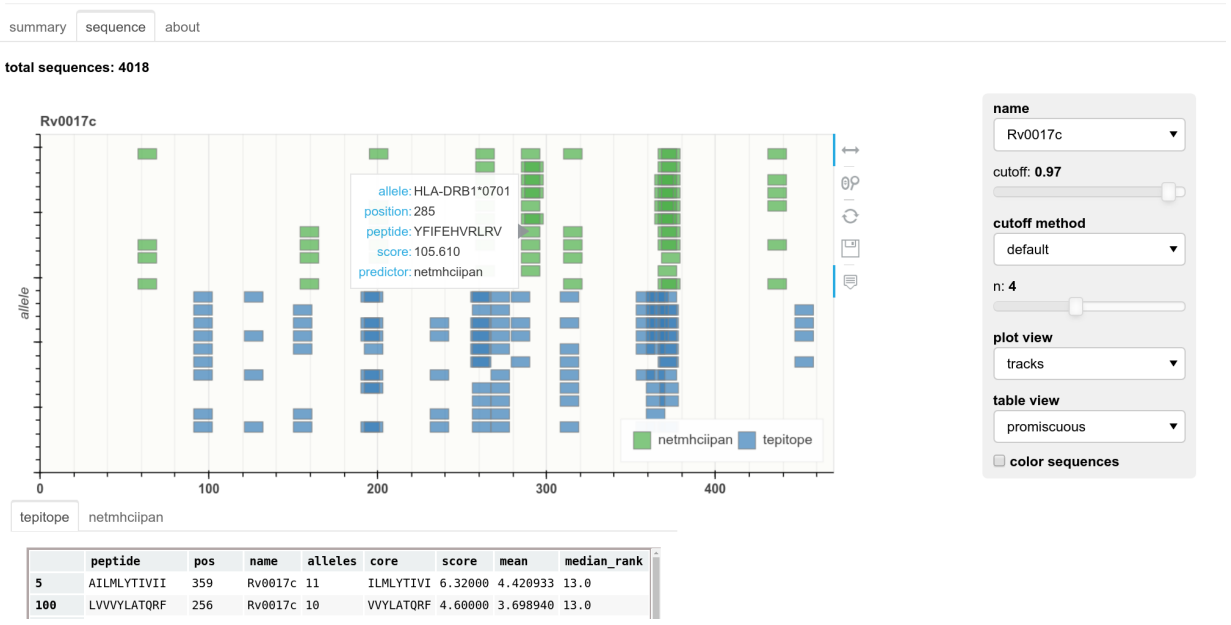

Figure 4: Web application showing results for a single protein sequence. Widgets can be used to select protein, cut-off levels and plot display.

Conclusions

This software provides a programmatic framework and command line interface for running multiple MHC binding prediction algorithms. This will be especially useful for performing high throughput calculations in many sequences and alleles. It is designed to scale for proteome scanning by allowing multiple processing threads to be used with any of the prediction methods. The API can also be easily applied to single sequences or small numbers of antigens. A web interface allows users to readily review results if they wish.

Availability and requirements

Project name: epitopepredict
Project home page: https://github.com/dmnfarrell/epitopepredict
Archived version: v0.5.0 (DOI: 10.5281/zenodo.4056421)
SciCrunch Identifier: SCR_019221
Operating system(s): Linux, Unix
Programming language: Python
Other requirements: biopython, pandas, numpy, matplotlib, scikit-learn, bokeh
License: GNU General Public License v 3.0
Any restrictions to use by non-academics: None

Funding

This work was supported by the Irish Department of Agriculture Food and the Marine grant 15/S/651 (NEXUSMAP). DF was previously funded under an Irish Research Council Postdoctoral Fellowship (GOIPD/2015/475) for part of this work. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgments

Thanks to Dr. Joseph Crispell for useful discussions on machine learning. Also thanks to Prof. Stephen Gordon for support during the development of this software.

Availability of data and materials

All computational work described here was implemented using Python. The code is provided as a Python package called *epitopepredict* under the Apache license. Extensive use was made of the IPython (Jupyter) notebook environment [30] in prototyping the codebase.
Documentation for users is available at http://epitopepredict.readthedocs.io

Installation

This software should be run on a Linux operating system. Ubuntu is recommended but most major distributions will work well. Windows is not supported. If using Windows or macOS (OS X), users can simply install Linux using virtual machine software such as Oracle VM VirtualBox (https://www.virtualbox.org). Software is then installed using the online documentation. The installation process is very simple, requiring only a single typed command. Externally used MHC binding prediction algorithms do need to be installed separately, these are all freely available.

Installing netMHCpan and netMHCIIpan

Due to license restrictions these programs must be installed separately. They are free for academic users. You can go to http://www.cbs.dtu.dk to fill in the forms that will give you access to the install file for the respective programs. The install instructions can then be found in the readme files when you untar the downloaded file e.g. netMHCpan-4.1.readme. There are four steps detailed and the process is relatively simple. Remember to test the software is working before you use it in epitopepredict.

References

1. Lundegaard C, Hoof I, Lund O, Nielsen M. State of the art and challenges in sequence based T-cell epitope prediction. *Immunome Res*. BioMed Central Ltd; 2010; doi: 10.1186/1745-7580-6-S2-S3.

2. Backert L, Kohlbacher O. Immunoinformatics and epitope prediction in the age of genomic medicine. *Genome Med*. Genome Medicine; 2015; doi: 10.1186/s13073-015-0245-0.

3. Domingos P. A few useful things to know about machine learning. *Commun ACM*. 2012; doi: 10.1145/2347736.2347755.

4. Chaves F a, Lee AH, Nayak JL, Richards K a, Sant AJ. The utility and limitations of current Web-available algorithms to predict peptides recognized by CD4 T cells in response to pathogen infection. *J Immunol*. 2012; doi: 10.4049/jimmunol.1103640.

5. Bremel RD, Homan EJ. An integrated approach to epitope analysis II: A system for proteomic-scale prediction of immunological characteristics. *Immunome Res*. BioMed Central Ltd; 2010; doi: 10.1186/1745-7580-6-8.

6. Paul S, Weiskopf D, Angelo MA, Sidney J, Peters B, Sette A. HLA class I alleles are associated with peptide-binding repertoires of different size, affinity, and immunogenicity. *J Immunol*. 2013; doi: 10.4049/jimmunol.1302101.

7. Schubert B, Lund O, Nielsen M. Evaluation of peptide selection approaches for epitope-based vaccine design. *Tissue Antigens*. 2013; doi: 10.1111/tan.12199.

8. Soria-Guerra RE, Nieto-Gomez R, Govea-Alonso DO, Rosales-Mendoza S. An overview of bioinformatics tools for epitope prediction: Implications on vaccine development. *J Biomed Inform*. Elsevier Inc.; 2015; doi: 10.1016/j.jbi.2014.11.003.

9. De Groot AS, Martin W. Reducing risk, improving outcomes: bioengineering less immunogenic protein therapeutics. *Clin Immunol*. Elsevier Inc.; 2009; doi: 10.1016/j.clim.2009.01.009.

10. Mckinney W: Pandas, Python Data Analysis Library. http://pandas.pydata.org/ (2015).

11. FriendFeed: Tornado: Python web framework and asynchronous networking library. https://www.tornadoweb.org/en/stable/

12. Bokeh Developers: Bokeh. https://bokeh.org/ (2020).

13. Zhang L, Chen Y, Wong H-S, Zhou S, Mamitsuka H, Zhu S. TEPITOPEpan: extending TEPITOPE for peptide binding prediction covering over 700 HLA-DR molecules. *PLoS One*. 2012; doi: 10.1371/journal.pone.0030483.

14. Sturniolo T, Bono E, Ding J, Raddrizzani L, Tuereci O, Sahin U, et al.. Generation of tissue-specific and promiscuous HLA ligand databases using DNA microarrays and virtual HLA class II matrices. *Nat Biotechnol*. 1999; doi: 10.1038/9858.

15. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al.. Scikit-learn: Machine Learning in Python. *J Mach Learn Res*. JMLR.org; 12:2825–302011;

16. O'Donnell TJ, Rubinsteyn A, Bonsack M, Riemer AB, Laserson U, Hammerbacher J. MHCflurry: Open-Source Class I MHC Binding Affinity Prediction. *Cell Syst*. Elsevier Inc.; 2018;

doi: 10.1016/j.cels.2018.05.014.

17. Jurtz V, Paul S, Andreatta M, Marcatili P, Peters B, Nielsen M. NetMHCpan-4.0: Improved Peptide–MHC Class I Interaction Predictions Integrating Eluted Ligand and Peptide Binding Affinity Data. *J Immunol*. 2017; doi: 10.4049/jimmunol.1700893.

18. Nielsen M, Justesen S, Lund O, Lundegaard C, Buus S. NetMHCIIpan-2.0 - Improved pan-specific HLA-DR predictions using a novel concurrent alignment and weight optimization training procedure. *Immunome Res*. 2010; doi: 10.1186/1745-7580-6-9.

19. Karosiene E, Rasmussen M, Blicher T, Lund O, Buus S, Nielsen M. NetMHCIIpan-3.0, a common pan-specific MHC class II prediction method including all three human MHC class II isotypes, HLA-DR, HLA-DP and HLA-DQ. *Immunogenetics*. 2013; doi: 10.1007/s00251-013-0720-y.

20. Nanni L, Lumini A. A new encoding technique for peptide classification. *Expert Syst Appl*. Elsevier Ltd; 2011; doi: 10.1016/j.eswa.2010.09.005.

21. Gök M, Özcerit AT. OETMAP: A new feature encoding scheme for MHC class i binding prediction. *Mol Cell Biochem*. 2012; doi: 10.1007/s11010-011-1000-5.

22. Kim Y, Sidney J, Buus S, Sette A, Nielsen M, Peters B. Dataset size and composition impact the reliability of performance benchmarks for peptide-MHC binding predictions. *BMC Bioinformatics*. 2014; doi: 10.1186/1471-2105-15-241.

23. Lindestam Arlehamn CS, McKinney DM, Carpenter C, Paul S, Rozot V, Makgotlho E, et al.. A Quantitative Analysis of Complexity of Human Pathogen-Specific CD4 T Cell Responses in Healthy M. tuberculosis Infected South Africans. *PLOS Pathog*. 2016; doi: 10.1371/journal.ppat.1005760.

24. Farrell D, Jones G, Pirson C, Malone K, Rue-Albrecht K, Chubb AJ, et al.. Integrated computational prediction and experimental validation identifies promiscuous T cell epitopes in the proteome of Mycobacterium bovis. *Microb Genomics*. 2016; doi: 10.1099/mgen.0.000071.

25. Santos AR, Pereira VB, Barbosa E, Baumbach J, Pauling J, Röttger R, et al.. Mature Epitope Density - A strategy for target selection based on immunoinformatics and exported prokaryotic proteins. *BMC Genomics*. BioMed Central Ltd; 2013; doi: 10.1186/1471-2164-14-S6-S4.

26. Doshi P. Covid-19: Do many people have pre-existing immunity? *BMJ*. 2020; doi: 10.1136/bmj.m3563.

27. Grifoni A, Weiskopf D, Ramirez SI, Mateus J, Dan JM, Rydyznski Moderbacher C, et al.. Targets of T cell responses to SARS-CoV-2 coronavirus in humans with COVID-19 disease and unexposed individuals. *Cell*. Elsevier Inc.; 2020; doi: 10.1016/j.cell.2020.05.015.

28. Mateus J, Grifoni A, Tarke A, Sidney J, Ramirez SI, Dan JM, et al.. Selective and cross-reactive SARS-CoV-2 T cell epitopes in unexposed humans. *Science (80-)*. 2020; doi: 10.1126/science.abd3871.

29. Farrell D, Gordon S V.. Epitopemap: A web application for integrated whole proteome epitope prediction. *BMC Bioinformatics*. 2015; doi: 10.1186/s12859-015-0659-0.

30. Project Jupyter: Jupyter Notebook. http://jupyter.org/ (2015). Accessed 2016 Jan 21.