Computational antigen discovery for eukaryotic pathogens using *Vacceed*

Stephen J. Goodswen[1], Paul J. Kennedy[2], John T. Ellis[1,*]

[1] School of Life Sciences, University of Technology Sydney (UTS), Ultimo, NSW, Australia.

[2] School of Software, Faculty of Engineering and Information Technology and the Centre for Artificial Intelligence, University of Technology Sydney (UTS), Ultimo, NSW, Australia.

Correspondence: John.Ellis@uts.edu.au

Running Head: Eukaryotic pathogen antigen discovery using Vacceed

Abstract

Bioinformatics programs have been developed that exploit informative signals encoded within protein sequences to predict protein characteristics. Unfortunately, there is no program as yet that can predict whether a protein will induce a protective immune response to a pathogen. Nonetheless, predicting those pathogen proteins most likely from those least likely to induce an immune response is feasible when collectively using protein characteristics that can be predicted. *Vacceed* is a computational pipeline that manages different standalone bioinformatics programs to predict various protein characteristics, which offer supporting evidence on whether a protein is secreted or membrane associated. A set of machine learning algorithms predicts the most likely pathogen proteins to induce an immune response given the supporting evidence. This chapter provides step by step descriptions of how to configure and operate *Vacceed* for a eukaryotic pathogen of the user's choice.

Key Words *Vacceed*, machine learning, *in silico* vaccine discovery, computational antigen discovery, eukaryotic pathogen

1. Introduction

Proteins sequences are not random assemblies of amino acids. There is a precise biological reason why one particular amino acid is connected to another, which ultimately contributes to a protein's distinctive characteristics [1]. Researchers, over the last two decades, have developed bioinformatics programs that exploit informative signals or patterns encoded within these amino acid sequences to predict protein characteristics. Examples of these characteristics are subcellular localization [2], presence and location of signal peptide cleavage sites [3], and transmembrane topology[4]. With respect to discovering protein vaccine candidates, no signal has yet been detected that helps predict a characteristic signifying a protein's contributing capacity to a protective immune response in a host. Consequently, the current computational antigen discovery aspiration is to distinguish those pathogen proteins most likely (referred to henceforth as positives) from those least likely (referred to henceforth as negatives) to induce an immune response.

Vacceed is the collective name for a configurable pipeline of linked bioinformatics programs, Perl scripts, R functions and Linux shell scripts [5]. It was inspired by the principles of reverse vaccinology [6], whereby antigen discovery starts *in silico* using the pathogen genome rather than the traditional culture-based method of cultivating and dissecting the pathogen itself. *Vacceed* has been designed to facilitate an automated, high-throughput computational approach to predict vaccine candidates against eukaryotic pathogens given protein sequences [7]. The pipeline uses various standalone bioinformatics programs to predict various protein characteristics. *Vacceed* is grounded on the underlying premise that there is an expected difference between the set of characteristics defining positives to those of negatives. These differences are typically not apparent to an observer and hence applying a rule-based approach to distinguish proteins is not feasible. Conversely, machine learning

(ML) has the capacity to detect obscure differences. *Vacceed* uses a set of ML algorithms trained on protein characteristics of known positives and negatives to distinguish if a yet to be classified protein is a positive or negative [8]. So far, *Vacceed* has been used in studies to predict vaccine candidates for *Neospora caninum* [9] and *Cystoisospora suis* [10].

This chapter provides step by step descriptions of how to configure and operate *Vacceed* for a eukaryotic pathogen of the user's choice. A prerequisite for pathogen choice, nonetheless, is a substantial representation of the pathogen's proteome in the form of quality protein sequences.

2. *Vacceed* core background information

Vacceed can be downloaded from: https://github.com/goodswen/vacceed/releases. The download package includes a comprehensive *Vacceed* User Guide and sample data. Note that *Vacceed* has been designed for a Linux operating system and has only been tested on Red Hat Enterprise Linux 7.5, but is expected to work on most Linux distributions.

Each data processing stage in the *Vacceed* pipeline is an independent resource, which is built from a central Linux shell script encapsulating all programs needed to perform specific but related tasks. Typical tasks include predicting a particular protein characteristic. By default, *Vacceed* uses seven bioinformatics programs to predict protein characteristics: SignalP 5.0 [11] (predicts presence and location of signal peptide cleavage sites using deep neural networks); WoLF PSORT 0.2 [12] and TargetP 1.1 [2] (predict subcellular localization); TMHMM 2.0 [4] (predicts transmembrane domains in proteins); Phobius 1.01 [13] (predicts transmembrane topology and signal peptides); DeepLoc 1.0 [14] (predicts eukaryotic protein subcellular localization using deep learning); and IEDB peptide-MHC binding predictors (MHCI version 2.17 and MHCII version 2.16.3) [15] (*see* **Note 1**). Observe that each of the

seven programs have specific version numbers on which *Vacceed* has been tested. There is no assurance older or newer program versions will work.

The most pertinent file from a user's perspective is a species configuration file in a header-key format (see Fig. 1). For example, [Resources] is the header, and 'name' is the key. Text following the '=' sign is configurable. A suggested convention is to have one configuration file for each target pathogen. *Vacceed* is started by entering only one command in a Linux Shell (or terminal) e.g. perl startup *xx*, where *xx* is a user specified code that links *Vacceed* to the target pathogen configuration file. No other commands are required.

Once *Vacceed* is started, each resource listed after the 'name' key is consecutively executed. Resource names can be in any order or even excluded with the exception of VALIDATE (*see* **Note 2**) and EVIDENCE (*see* **Note 3**), which must always be the first and last in the list, respectively. Any key in the configuration file can be used as a variable replacement in the rest of the configuration file. That is, a '$' character preceding a word denotes a variable e.g. $work_dir is replaced by '$HOME/vacceed' throughout the configuration file on execution.

Typical *Vacceed* run times are dependent on various factors including numbers of proteins to process, programs to execute (resources), computer processors (cores), and the amount of memory. For example, a test with 500 proteins processed through all resources completed in 3 hours, 21 minutes, and 17 seconds using Red Hat Enterprise Linux Workstation release 7.5, 64 bit kernel, and 32 MB memory with 8 cores; however, the same test without the resources MHCI and MHCII completed in 23 minutes and 54 seconds. *Vacceed* takes advantage of multi-core processors. By default, the proteins to process are split into subsets by the number of cores and then each subset is processed in parallel.

3. Methods

3.1 Running Vacceed with sample data

The *Vacceed* installation provides sample data comprising a small collection of *Toxoplasma gondii* proteins as input. The purpose of this section is to test the *Vacceed* installation.

1. Install *Vacceed* (*see* **Note 4**).
2. Edit the species configuration file 'toxoplasma.ini' located in the directory <*install_dir*>/vacceed/start/config_dir (where <*install_dir*> is the directory in which *Vacceed* was installed). Under the [Resources] header, remove MHCI and MHCII (*see* **Note 5**).
3. Under the [Main] header, change the current path assigned to work_dir to *install_dir*/vacceed/.
4. Under the [Main] header, assign an appropriate e-mail address to email_url
5. In a command-line terminal, change directory to *install_dir*/vacceed/start.
6. Enter the command: **perl startup tg**
7. An e-mail will automatically be sent either when the pipeline is successfully completed or immediately when an error occurs. A log file is attached to the e-mail providing details of success or failure (*see* **Note 6**).
8. If successful, the main output file called 'vaccine_candidates' is created in the directory *install_dir*/vacceed/toxoplasma/proteome. This file contains a list of all processed proteins ranked on average ML scores (*see* Fig 2. and **Note 7**).

3.2 Running *Vacceed* with user provided data

Once the *Vacceed* installation has been successfully tested, *Vacceed* can be configured and operated for a eukaryotic pathogen of the user's choice. *Neospora caninum* is used here for demonstration purposes.

1. Collect *all* known protein sequences of the target pathogen into one file (*see* **Note 8**). The sequences must be in a FASTA format with a sequence identifier in the following layout: >*xx* |protein Identifier (ID)| text (optional), where *xxx* can be any characters e.g. 'tr' or 'sp' as per UniProt identifiers.
2. Copy file from step #1 into *install_dir*/vacceed/start/proteome
3. Copy the entire template_species directory to a user-named directory, e.g., neospora.
4. Copy the species configuration file 'toxoplasma.ini' located in the directory *install_dir*/vacceed/start/config_dir to 'neospora.ini'.
5. Add a new line to startup.ini located in *install_dir*/vacceed/start/:
   nc< Neospora caninum <pipeline<neospora.ini< *install_dir*/vacceed/start/config_dir
6. Edit neospora.ini to match the following:
   work_dir="*install_dir*/vacceed"
   species_dir="neospora"
   email_url="your_email@address" (user e-mail address)
   proteome_fasta="proteome.fasta" (protein sequence file as per step #1)
   prot_id_prefix="*xxx*" (needs to match the sequence identifier as per step #1)
7. Modify the [Resources] in neospora.ini, if required. That is, remove any resource names between VALIDATE and EVIDENCE that are not required e.g. MHCI and MHCII.
8. Change directory to *install_dir*/vacceed/start in a command-line terminal.
9. Enter the command: **perl startup nc** (where 'nc' is as per step #5).
10. Check results in 'vaccine_candidates' in *install_dir*/vacceed/neospora/proteome

3.3 Creating pathogen specific training data

Training data here is essentially the collection of predicted evidence (referred to henceforth as evidence profiles) from the seven bioinformatics programs for those proteins known to be

positive or negative. A training data file called 'train_profiles' is provided with the *Vacceed* package as part of the *T. gondii* sample data (*see* **Note 9**). A previous study [8] tested *Vacceed* with different evidence profiles compiled from different eukaryotic species. It concluded that there is no fundamental difference in evidence profile patterns, e.g., a model trained on one species can be used to classify proteins from another. This is because the bioinformatics programs are designed or ML trained for eukaryotes in general. Therefore, the creation of a pathogen specific training dataset is not a mandatory step. However, an ideal training dataset is one that contains the greatest variety of evidence profiles (*see* **Note 10**) irrespective of the source species, e.g., quality and variety are indisputably the most important factors that impact the accuracy of ML algorithms [8]. A new or amended training file is recommended under any of the following circumstances: a bioinformatics program is upgraded, i.e., it has improved accuracy; experimentally proved immunogenic proteins become available; and a new prediction program is added (see Section 3.5).

1. Collect as many proteins as possible for the target species that are known to induce an immune response in the relevant host. The proteins will represent the 'positives' for the training file (*see* **Note 11**).
2. Collect proteins that *do not* induce an immune response. These proteins will represent the 'negatives' (*see* **Note 12**).
3. Create a file (e.g., positives.fasta) containing the positive sequences in a FASTA format.
4. Create a file (e.g., negatives.fasta) containing the negative sequences in a FASTA format.
5. Copy both FASTA files into *install_dir*/vacceed/start/proteome
6. Copy the entire template_species directory to a user-named directory, e.g., training.
7. Copy 'toxoplasma.ini' to 'train.ini'.

8. Add a new line to startup.ini located in *install_dir*/vacceed/start/:
   train< Neospora caninum <pipeline<train.ini< *install_dir*/Vacceed/start/config_dir
9. Edit train.ini to match the following:
   work_dir="*install_dir*/vacceed"
   species_dir="training"
   email_url="your_email@address" (user e-mail address)
   proteome_fasta="positives.fasta" (as per step #3)
   prot_id_prefix="*xxx*" (needs to match the sequence identifier)
11. Modify [Resources] in train.ini if required e.g. remove any resource not installed or required.
12. Change directory to *install_dir*/vacceed/start in a command-line terminal.
13. Enter the command: **perl startup train** (where 'train' is as per step #8).
14. Copy the file 'evidence_profiles' from *install_dir*/vacceed/training/pipeline/evidence/output to *install_dir*/vacceed/training/pipeline/evidence/training_files
15. Rename evidence_profiles to a user-defined name e.g. neospora_profiles
16. Add ',YES' to the end of each row in the new training file (exclude the first row). The 'YES' is the required target label for the positives.
17. Edit train.ini to match the following:
    proteome_fasta="negatives.fasta" (as per step #4)
18. Change directory to *install_dir*/vacceed/start in a command-line terminal.
19. Enter the command: **perl startup train**
20. Add ',NO' (i.e., the required target label for the negatives) to the end of each row in evidence_profiles in *install_dir*/vacceed/training/pipeline/evidence/output

21. Append the entire contents of the amended evidence_profiles (except first row) to the new training file, e.g., neospora_profiles.
22. Copy new training file to *install_dir*/vacceed/<new species>/evidence/training_files where <new species> is the directory created for the target species, e.g., neospora.
23. Edit the species configuration file, e.g., neospora.ini and change the value of the train_file key under header [EVIDENCE] to the new training file, e.g., neospora_profiles (*see* **Note 13**).
24. The new training data should be evaluated with techniques such as k-fold cross validation (*see* **Note 14**) and the ML algorithm parameters tweaked to improve performance (*see* **Note 15**).

3.4 Creating MHCI and MHCII training data

This section is only applicable when using resources MHCI and/or MHCII ***and*** the target pathogen host is **not** human. By default, *Vacceed* uses human alleles (e.g. HLA-A*01:01) for peptide-MHC binding predictions. The following describes steps required to setup MHCI for a host other than human, e.g., mouse.

1. Follow steps #1 to #5 from Section 3.3.
2. Create a file (e.g. mouse_mchI_alleles) in a comma delimited format containing all required mouse alleles and peptide lengths e.g. H-2-IAb,8 where each 'allele,length' is on a separate line (*see* **Note 16**).
3. Copy the entire template_species directory to a user-named directory, e.g., mouse.
4. Copy mouse_mchI_alleles to < *install_dir*/Vacceed/mouse/pipeline/mhci/alleles
5. Copy 'toxoplasma.ini' to 'mouse.ini'.
6. Add a new line to startup.ini located in *install_dir*/vacceed/start/:
   m< mouse <pipeline<mouse.ini< *install_dir*/Vacceed/start/config_dir
7. Edit mouse.ini to match the following:

work_dir="*install_dir*/vacceed"

species_dir="mouse"

email_url="your_email@address" (user e-mail address)

proteome_fasta="positives.fasta"

prot_id_prefix="*xxx*" (needs to match the sequence identifier)

allele_file="mouse_mchI_alleles" (located under the resource [MHCI_files])

8. Modify [Resources] in mouse.ini to 'name=VALIDATE,MHCI'
9. Change directory to *install_dir*/vacceed/start in a command-line terminal.
10. Enter the command: **perl startup m** (where 'm' is as per step #6).
11. Copy mhci_ml.txt from *install_dir*/vacceed/mouse/pipeline/mhci/output from *install_dir*/vacceed/mouse/pipeline/mhci/training_files
12. Rename mhci_ml.txt to a user-defined name e.g. mouse_mhci_ml.txt
13. Add ',YES' to the end of each row in the new training file (exclude the first row)
14. Edit mouse.ini to match the following:

    proteome_fasta="negatives.fasta"
15. Change directory to *install_dir*/vacceed/start in a command-line terminal.
16. Enter the command: **perl startup m**
17. Add ',NO' to the end of each row in mhci_ml.txt in *install_dir*/vacceed/mouse/pipeline/mhci/output
18. Append the entire contents of the amended mhci_ml.txt (except first row) to the new training file, e.g., mouse_mhci_ml.txt.
19. Copy new training file to *install_dir*/vacceed/<new species>/mhci/training_files where <new species> is the directory created for the target species, e.g., new_mouse.

20. Edit the species configuration file, e.g., new_mouse.ini, and change the value of the train_file key under header [MHCI_files] to the new training file, e.g., mouse_mhci_ml.txt
21. Repeat the steps above to create a MHCII training file, but change mhci to mhcii (*see* **Note 17**).

3.5 Add a new resource

New programs to predict protein characteristics will inevitably be developed in the future. This section describes how to incorporate a new program into *Vacceed*, which essentially is adding a new resource with the goal of extracting relevant evidence from the new program output to append to evidence profiles.

1. Install and test new program with sample data to ensure it runs successfully from any directory (*see* **Note 18**).
2. Determine the input requirements and the output format of new program.
3. Add a new resource name, e.g., program_Z in an appropriate configuration file:
   [Resources]
   name=VALIDATE,WOLF,TMHMM,**PROGRAM_Z**,EVIDENCE
4. Add a new section to the same configuration file. The easiest way to do this is to copy an existing resource and amend accordingly (see Fig. 3). The texts highlighted in red are the only parts expected to be changed.
5. Create a new directory in *install_dir*/vacceed/*new_species*/pipeline using the same name as the new resource (but in lowercase), e.g., program_z.
6. Create two directories called 'output' and 'scripts' in the program_z directory.
7. Copy 'template_resource_script' from *install_dir*/vacceed/*new_species*/pipeline/common_programs to *install_dir*/vacceed/*new_species*/pipeline/program_z

8. Rename ‘template_resource_script’ to a user-named file e.g. program_z_script (*see* **Note 19**)
9. Amend program_z_script where it states << Add new programs here >>, e.g.,

```
echo "script_step=\">> executing program_z\"" >> $script_dir/script$chr_no
echo "program_z $required_input $out_dir" >> $script_dir/script$chr_no || error_exit
```

Where $required_input is the input as determined in step #2.

10. A generic Perl script called ‘get_evidence.pl’ (located in: *install_dir*/vacceed/*new_species*/pipeline/common_programs) can be amended accordingly to extract the relevant evidence from the program_z output file (*see* **Note 20**). Alternatively, any programming language can be used to write a program to extract evidence. In such a case, the program name would need to replace ‘get_evidence.pl’ in program_z_script. Regardless of the extraction program, evidence needs to be saved in a user-named file with the suffix ‘_evd’, e.g., programz_evd in the directory *install_dir*/vacceed/*new_species*/pipeline/evidence/output.

4. Notes

1. The bioinformatics programs are third-party and are not part of the *Vacceed* package. Furthermore, installation steps for the third-party programs are not described in this chapter. Most of the programs provide a **ReadMe** file with instructions. Even so**,** these installations are still a challenging aspect to preparing *Vacceed* ready for use. It is highly recommended to seek the help of an administrator or an experienced Linux user.
2. *Vacceed* checks to see if a protein sequence contains invalid letters, e.g., characters other than [ACDEFGHIKLMNPQRSTVWY].

3. *Vacceed* collates relevant, predicted protein characteristics (typically in the form of numerical values) into one file called evidence_profiles, i.e., contents of all files with the extension '_evd' in the evidence/output directory are combined as columns into evidence_profiles.
4. Ensure that each third-party program runs successfully before testing *Vacceed.*

5 The computation of peptide-MHC predictions takes less than a few minutes depending on the computer environment to run the test when both MHCI and MHCII are removed. Furthermore, MHCI and MHCII predictions on the whole are not accurate [16] (particularly MHCII [17]) and only marginally contributed to the *Vacceed* end result when tested with the *T. gondii* sample data [8].

6 If *Vacceed* fails with the test data then it will inevitably fail with any other data. The expected reason for the failure is installation issues of one or more of the third-party programs (*see* **Note 4**). The log file may give clues as to which third-party program(s) is the culprit.

7 The ML algorithms used are listed in the configuration file under the header [Evidence] and the key 'algorithms'.

8 It is recommended that *all* known pathogen proteins are processed irrespective of protein name or expected function. This allows for an unbiased approach.

9 This training file contains 475 positives of mainly *T. gondii proteins* (nine are *N. caninum*). A small selection of these proteins are known to induce an immune response, but most are proteins predicted to be membrane-associated or secreted, i.e., proteins exposed to the immune system. There are 501 *T. gondii* proteins representing negatives, which were defined by the protein's predicted sub-cellular location, i.e., neither membrane-associated nor secreted.

10 Variety, in this instance from a ML perspective, is having a generalised selection of proteins in the training file that are representative of all conceivable types of positive and negative proteins, e.g., with a limited selection, a ML algorithm may not generalize to evidence profiles not seen when it was learning (i.e., poorly predicts when given new data).

11 Finding training proteins for most species is not a trivial task. The expectation is that a thorough search of the literature will be required. Even then, there may still be an inadequate number of examples to create a training file. A suggested compromise is to use positive proteins from a closely related organism or proteins 'expected' to induce or not induce an immune response. For instance, use proteins known to be exposed to the immune system (e.g., membrane associated or secreted proteins) for positives and non-exposed proteins (e.g., proteins normally located in the interior of the organism) as negatives.

12 A drawback for collecting negative examples is that a protein cannot definitively be defined a negative unless it has been explicitly tested in a laboratory.

13 The same proteins should never be used for training and evaluation. This would introduce biased results. Typically, the proteins are randomly divided into two sets. One set containing the majority of data e.g. 80% for training. The other set (e.g., 20%) used to evaluate the trained model's performance.

14 *k*-Fold cross-validation is a resampling statistical method used to estimate the performance of ML models. The '*k*' refers to the number of groups that a given data sample is to be split, e.g., 10-fold cross-validation indicates the sample data is split into 10 groups. One group in turn is used as a test dataset and the remaining groups used for training. The average of the *k* evaluation scores provides an indication of

how the model is expected to perform when used to make predictions on data not used during model training.

15 The distributed version of *Vacceed* is configured to run ML algorithms via R functions contained in packages. The algorithms are executed using Rscript. There are three R functions in *install_dir*/vacceed/<new species>/evidence/ that encapsulate the relevant command for each algorithm: <al>_wrapper.R, <al>_runPred.R, and <al>_makePred.R, where <al> is the algorithm abbreviation. Parameters to fine tune the algorithms can be modified in <al>_makePred.R e.g parameters 'ntree' and/or 'mtry' in rf_makePred.R, where rf = random forest, ntree = number of decision trees, and 'mtry' = number of variables to try at each split in the decision tree.

16 Run the following command to see available class I alleles: ./src/predict_binding.py IEDB_recommended mhc (only listed alleles can be used).

17 Run the following command to see available class II alleles: python mhc_II_binding.py allele (only listed alleles can be used).

18 May need to append new program location to the PATH variable.

19 This is a template script only and will need to be edited appropriately to suit the new program. There are user comments denoted by a '#' symbol, but a familiarity with Linux scripting is expected.

20 Amending get_evidence.pl requires experience in writing Perl scripts. Reading step #8 under the section 'Adding a new resource' in the Vacceed User Guide may prove useful when amending get_evidence.pl.

Acknowledgements

SJG gratefully acknowledges Zoetis (Pfizer) Animal Health for funding the development of *Vacceed* through a PhD scholarship.

References

1. Lee D, Redfern O, Orengo C (2007) Predicting protein function from sequence and structure. Nature Reviews Molecular Cell Biology 8 (12):995-1005. doi:10.1038/nrm2281
2. Emanuelsson O, Brunak S, von Heijne G, Nielsen H (2007) Locating proteins in the cell using TargetP, SignalP and related tools. Nature Protocols 2 (4):953-971. doi:10.1038/nprot.2007.131
3. Petersen TN, Brunak S, von Heijne G, Nielsen H (2011) SignalP 4.0: discriminating signal peptides from transmembrane regions. Nature Methods 8 (10):785-786. doi:10.1038/nmeth.1701
4. Krogh A, Larsson B, von Heijne G, Sonnhammer ELL (2001) Predicting transmembrane protein topology with a hidden Markov model: Application to complete genomes. Journal of Molecular Biology 305 (3):567-580. doi:10.1006/jmbi.2000.4315
5. Goodswen SJ, Kennedy PJ, Ellis JT (2014) Vacceed: a high-throughput in silico vaccine candidate discovery pipeline for eukaryotic pathogens based on reverse vaccinology. Bioinformatics 30 (16):2381-2383. doi:10.1093/bioinformatics/btu300
6. Rappuoli R (2000) Reverse vaccinology. Current Opinion in Microbiology 3 (5):445-450. doi:10.1016/s1369-5274(00)00119-3
7. Goodswen SJ, Kennedy PJ, Ellis JT (2013) A guide to in silico vaccine discovery for eukaryotic pathogens. Briefings in Bioinformatics 14 (6):753-774. doi:10.1093/bib/bbs066
8. Goodswen SJ, Kennedy PJ, Ellis JT (2013) A novel strategy for classifying the output from an in silico vaccine discovery pipeline for eukaryotic pathogens using machine learning algorithms. BMC Bioinformatics 14. doi:10.1186/1471-2105-14-315
9. Goodswen SJ, Kennedy PJ, Ellis JT (2017) On the application of reverse vaccinology to parasitic diseases: a perspective on feature selection and ranking of vaccine candidates. International Journal for Parasitology 47 (12):779-790. doi:10.1016/j.ijpara.2017.08.004
10. Palmieri N, Shrestha A, Ruttkowski B, Beck T, Vogl C, Tomley F, Blake DP, Joachim A (2017) The genome of the protozoan parasite Cystoisospora suis and a reverse vaccinology approach to identify vaccine candidates. International Journal for Parasitology 47 (4):189-202. doi:10.1016/j.ijpara.2016.11.007
11. Armenteros JJA, Tsirigos KD, Sonderby CK, Petersen TN, Winther O, Brunak S, von Heijne G, Nielsen H (2019) SignalP 5.0 improves signal peptide predictions using deep neural networks. Nature Biotechnology 37 (4):420-+. doi:10.1038/s41587-019-0036-z
12. Horton P, Park KJ, Obayashi T, Fujita N, Harada H, Adams-Collier CJ, Nakai K (2007) WoLF PSORT: protein localization predictor. Nucleic Acids Research 35:W585-W587. doi:10.1093/nar/gkm259
13. Kall L, Krogh A, Sonnhammer ELL (2004) A combined transmembrane topology and signal peptide prediction method. Journal of Molecular Biology 338 (5):1027-1036. doi:10.1016/j.jmb.2004.03.016
14. Armenteros JJA, Sonderby CK, Sonderby SK, Nielsen H, Winther O (2017) DeepLoc: prediction of protein subcellular localization using deep learning. Bioinformatics 33 (21):3387-3395. doi:10.1093/bioinformatics/btx431
15. Vita R, Zarebski L, Greenbaum JA, Emami H, Hoof I, Salimi N, Damle R, Sette A, Peters B (2010) The Immune Epitope Database 2.0. Nucleic Acids Research 38:D854-D862. doi:10.1093/nar/gkp1004
16. Bui HH, Sidney J, Peters B, Sathiamurthy M, Sinichi A, Purton KA, Mothe BR, Chisari FV, Watkins DI, Sette A (2005) Automated generation and evaluation of specific MHC binding predictive tools: ARB matrix applications. Immunogenetics 57 (5):304-314. doi:10.1007/s00251-005-0798-y
17. Wang P, Sidney J, Dow C, Mothe B, Sette A, Peters B (2008) A systematic assessment of MHC class II peptide binding predictions and evaluation of a consensus approach. Plos Computational Biology 4 (4). doi:10.1371/journal.pcbi.1000048

Figures

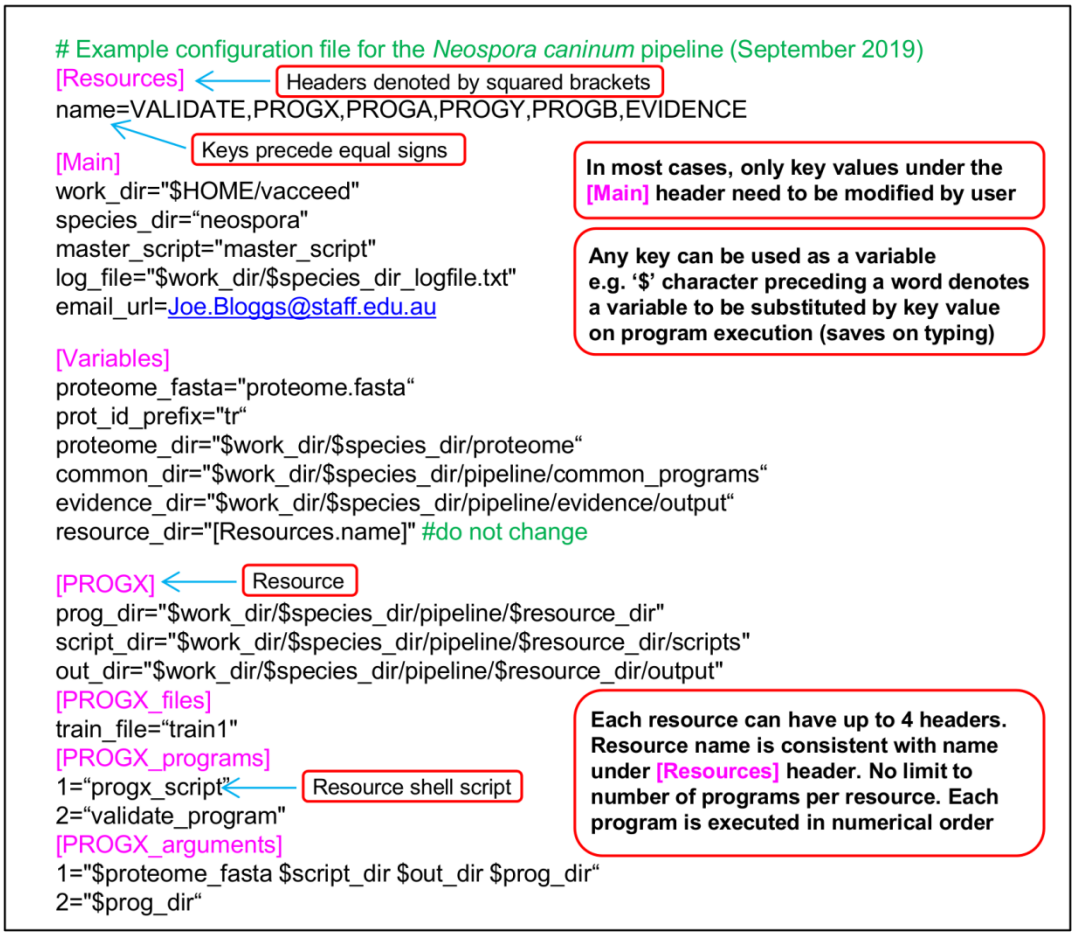
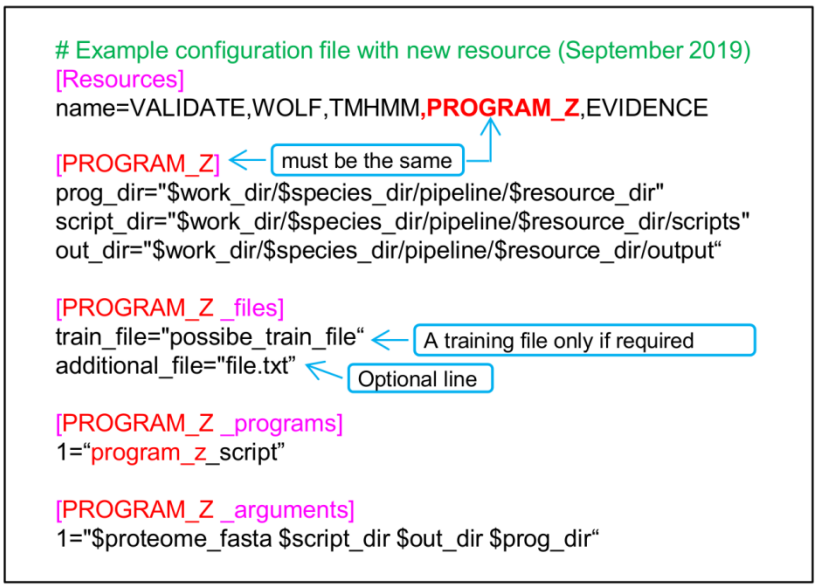


Fig. 1 Extract from a species configuration file

```
#ID,ada,knn,nb,nn,rf,svm,average_ML_score
BBOV_IV006420,1.000,1.000,1.000,1.000,1.000,0.994,0.999
BBOV_II001970,1.000,1.000,1.000,1.000,1.000,0.983,0.997
BBOV_III005590,0.984,0.667,0.883,0.800,0.604,0.872,0.801
BBOV_I004490,0.403,0.333,0.980,0.200,0.391,0.202,0.418
BBOV_IV002290,0.427,0.667,0.000,0.000,0.463,0.141,0.283
BBOV_III001400,0.015,0.333,0.764,0.200,0.255,0.121,0.281
BBOV_III011900,0.000,0.000,0.000,0.000,0.001,0.007,0.001
```

Fig. 2 Extract from main *Vacceed* output file 'vaccine_candidates'

Where ID = protein identifier, ada = adaptive boosting, knn = *k*-nearest neighbour classifier, nb = Naive Bayes classifier, nn = neural network, rf = random forest, and svm = support vector machines. vaccine_candidates is a comma delineated file containing an ordered list of all machine learning (ML) algorithm scores for each protein processed (seven in this instance). Each ML algorithm generates probabilities that the YES and NO classifications are correct, but only YES probabilities are displayed in the output. The 'average ML score' for each protein is the average probabilities of the YES classifications. The list order is descending based on 'average ML score' value. An appropriate threshold value (e.g., 0.5) can be compared to the average ML score to determine the relevant class, positive or negative.



434

435 **Fig. 3 Example of new resource added to species configuration file**

436