pVACtools: A Computational Toolkit to Identify and Visualize Cancer Neoantigens

Jasreet Hundal1, Susanna Kiwala1, Joshua McMichael1, Christopher A. Miller1,2,3, Huiming Xia1, Alexander T. Wollam1, Connor J. Liu1, Sidi Zhao1, Yang-Yang Feng1, Aaron P. Graubert1, Amber Z. Wollam1, Jonas Neichin1, Megan Neveau1, Jason Walker1, William E. Gillanders3,4, Elaine R. Mardis5, Obi L. Griffith1,2,3,6, and Malachi Griffith1,2,3,6

ABSTRACT

Identification of neoantigens is a critical step in predicting response to checkpoint blockade therapy and design of personalized cancer vaccines. This is a cross-disciplinary challenge, involving genomics, proteomics, immunology, and computational approaches. We have built a computational framework called pVACtools that, when paired with a well-established genomics pipeline, produces an end-to-end solution for neoantigen characterization. pVACtools supports identification of altered peptides from different mechanisms, including point mutations, in-frame and frameshift insertions and deletions, and gene fusions. Prediction of peptide:MHC binding is accomplished by supporting an ensemble of MHC Class I and II binding algorithms within a framework designed to facilitate the incorporation of additional algorithms. Prioritization of predicted peptides occurs by integrating diverse data, including mutant allele expression, peptide binding affinities, and determination whether a mutation is clonal or subclonal. Interactive visualization via a Web interface allows clinical users to efficiently generate, review, and interpret results, selecting candidate peptides for individual patient vaccine designs. Additional modules support design choices needed for competing vaccine delivery approaches. One such module optimizes peptide ordering to minimize junctional epitopes in DNA vector vaccines. Downstream analysis commands for synthetic long peptide vaccines are available to assess candidates for factors that influence peptide synthesis. All of the aforementioned steps are executed via a modular workflow consisting of tools for neoantigen prediction from somatic alterations (pVACseq and pVACfuse), prioritization, and selection using a graphical Web-based interface (pVACviz), and design of DNA vector-based vaccines (pVACvector) and synthetic long peptide vaccines. pVACtools is available at http://www.pvactools.org.

Introduction

The increasing use of cancer immunotherapies has spurred interest in identifying and characterizing predicted neoantigens encoded by a tumor's genome. The facility and precision of computational tools for predicting neoantigens have become increasingly important (1), and several resources aiding in these efforts have been published (2–4). Typically, these tools start with a list of somatic variants [in Variant Call Format (VCF) or other formats such as Browser Extensible Data (BED)] with annotated protein changes and predict the strongest MHC-binding peptides (8–11-mer for Class I MHC and 13–23-mer for Class II) using one or more prediction algorithms (5–7). The predicted neoantigens are then filtered and ranked based on defined metrics, including sequencing read coverage, variant allele fraction (VAF), gene expression, and differential binding compared with the wild-type peptide (agretropicity index score; ref. 8). However, of the prediction tools overviewed in Supplementary Table S1 [Vaxrank (3), MuPeXI (2), CloudNeo (4), FRED2 (9), Epi-Seq (8), and ProTECT (10)], most lack key functionality, including predicting neoantigens from gene fusions, aiding optimized vaccine design for DNA cassette vaccines, and explicitly incorporating nearby germline or somatic alterations into the candidate neoantigens (11). In addition, none of these tools offer an intuitive graphical user interface for visualizing and efficiently selecting the most promising candidates—a key feature for facilitating involvement of clinicians and other researchers in the process of neoantigen prediction and evaluation.

To address these limitations, we created a comprehensive and extensible toolkit for computational identification, selection, prioritization, and visualization of neoantigens—pVACtools—which facilitates each of the major components of neoantigen identification and prioritization. This computational framework can be used to identify neoantigens from a variety of somatic alterations, including gene fusions and insertion/deletion frameshift variants, both of which potentially create strong immunogenic neoantigens (12–14). pVACtools can facilitate both MHC Class I and II predictions and provides an interactive display of predicted neoantigens for review by the end user.
Materials and Methods

The Cancer Genome Atlas data preprocessing

Aligned (build GRCh38) tumor and normal BAMs from BWA (ref. 15; version 0.7.12-r1039) as well as somatic variant calls from VarScan2 (refs. 16, 17; in VCF format) were downloaded from the Genomic Data Commons (GDC, https://gdc.cancer.gov/). Because the GDC does not provide germline variant calling for The Cancer Genome Atlas (TCGA) data, we used GATK’s (18) HaplotyperCaller to perform germline variant calling using default parameters. These calls were refined using VariantRecalculator in accordance with GATK Best Practices (19). Somatic and germline missense variant calls from each sample were then combined using GATK’s CombineVariants, and the variants were subsequently phased using GATK’s ReadBackedPhasing algorithm.

Phased Somatic VCF files were annotated with RNA depth and expression information using VATools (http://vatools.org). We restricted our analysis to only consider “PASS” variants in these VCFs, as these are higher confidence than the raw set, and the variants were annotated using the “--pick” option in VEP (Ensembl version 88).

Existing in silico HLA typing information was obtained from The Cancer Immunome Atlas (TCIA) database (20).

Neoantigen prediction

The VEP-annotated VCF files were then analyzed with pVACseq using all eight Class I prediction algorithms for neoantigen peptide lengths 8–11. The current MHC Class I algorithms supported by pVACseq are NetMHCpan (21), NetMHC (7, 21), NetMHCcons (22), PickPocket (23), SMMPMBEC (25), MHCflurry (26), and MHChydrus (27). The four MHC Class II algorithms that are supported are NetMHCIIPan (28), SMMalign (29), NNalign (30), and MHChydrus (31). For the demonstration analysis, we limited our prediction to only MHC Class I alleles due to availability of HLA typing information from TCGA, though binding predictions for Class II alleles can also be generated using pVACtools.

Ranking of neoantigens

To help prioritize neoantigens, a rank is assigned to all neoantigens that pass initial filters. Each of the following four criteria is assigned a rank-ordered value (where the best is assigned rank = 1):

\[
B = \text{Rank of binding affinity} \\
F = \text{Rank of fold change between mutant and wild-type alleles} \\
M = \text{Rank of mutant allele expression, calculated as (rank of gene expression \times rank of mutant allele RNA variant allele fraction)} \\
D = \text{Rank of DNA variant allele fraction}
\]

A final ranking is based on a score obtained from combining these values: \(B + F + (M \times 2) + (D/2)\).

Whereas agretopicity is considered in ranking, we do not recommend filtering candidates solely based on mutant versus wild-type binding value (default fold change value is 0). A previous study on the binding properties of known neoantigen candidates showed that most changes reside in T-cell receptor contact regions and not in anchor residues (32). To help end users make an informed decision while selecting candidates, we also provide the position of the variant in the final report. We do not rely on hard rules about “anchor positions” (e.g., position 2 and C-terminal) because these may vary by MHC allele and the peptide length. As we gain more empirical results and validation data, we hope that future versions of pVACtools may include more options for prioritizing based on these factors.

The rank is not meant to be a definitive metric of peptide suitability for vaccines, but was designed to be a useful first step in the peptide selection process.

Implementation

pVACtools is written in Python3. The individual tools are implemented as separate command line entry points that can be run using the “pvacseq,” “pvacfuse,” “pvacvector,” “pvacapi,” and “pvacviz” commands for each respective tool. pVACapi is required to run pVACvizr so both the “pvacapi” and “pvacviz” commands need to be executed in separate terminals.


Multithreading and parallelization

As many prediction algorithms are CPU intensive, pVACseq, pVACfuse, and pVACvector support using multiple cores to improve runtime. Using this feature, calls to the Immune Epitope Database (IEDB) and other prediction algorithms are made in parallel over a user-defined number of processes.

The pypm-pypi package was used to add support for parallel processing. The number of processes is controlled by the --n-threads parameter.

pVACseq

For pVACseq, the pycvf package is first used for parsing the input VCF file and extracting information about the supported missense, in-frame insertion, in-frame deletion, and frameshift variants into TSV format.

This output is then used to determine the wild-type or peptide sequence by extracting a region around the somatic variant according to the --peptide-sequence-length parameter. The user’s amino acid change is incorporated in this peptide sequence to determine the mutant peptide sequence. For frameshift variants, the new downstream protein sequence calculated by VEP is reported from the variant position onward. The number of downstream amino acids to include is controlled by the --downstream-sequence-length parameter. If a phased VCF with proximal variants is provided, proximal missense variants that are in phase with the somatic variant of interest are incorporated into the mutant and wild-type peptide sequences, as appropriate. The mutant and wild-type sequences are stored in a FASTA file. The FASTA file is then submitted to the individual prediction algorithms for binding affinity predictions. For algorithms included in the IEDB, we use either the IEDB API or a standalone installation, if an installation path is provided by the user (--iedb-install-directory). The mhcflurry and mhcnuggets packages are used to run the MHCflurry (26) and MHChydrus (27) prediction algorithms, respectively.

The predicted mutant antigens are then parsed into a TSV report format, and, for each mutant antigen, the closest wild-type antigen is
pVACtools: Toolkit for Tumor Neoantigen Characterization

determined and reported. Predictions for each mutant antigen/neoantigen from multiple algorithms are aggregated into the "best" (lowest) and median binding scores. The resulting TSV is processed through multiple filtering steps (as described below in Results: Comparison of filtering criteria): (i) Binding filter: this filter selects the strongest binding candidates based on the mutant binding score and the fold change (wild-type score/mutant score). Depending on the --top-score-metric parameter setting, this filter is applied to either the median score across all chosen prediction algorithms (default) or the best (lowest) score among the chosen prediction algorithms. (ii) Coverage filter: this filter accepts VAF and coverage information from the tumor DNA, tumor RNA, and normal DNA if these values are available in the input VCF. (iii) Transcript-support-level (TSL) filter: this filter evaluates each transcript's support level if this information was provided by VEP in the VCF. (iv) Top-score filter: the filter picks the top mutant peptide for each variant using the binding affinity as the determining factor. This filter is implemented to only select the best candidate from among multiple candidates that could result from a single variant due to different peptide lengths, variant registers, transcript sequences, and HLA alleles. The result of these filtering steps is reported in a filtered report TSV. The remaining neoantigens are then annotated with cleavage site and stability predictions by NetChop and NetMHCStabPan, respectively, and a relative rank (as described in the Materials and Methods: Ranking of neoantigens) is assigned. The rank-ordered final output is reported in a condensed file. The pandas package is used for data management while filtering and ranking the neoantigen candidates.

pVACvector

When running pVACvector with a pVACseq output file, the original input VCF must also be provided (--input-vcf parameter). The VCF is used to extract a larger peptide sequence around the target neoantigen (length determined by the --input-n-mer parameter). Alternatively, a list of target peptide sequences can be provided in a FASTA file. The set of peptide sequences is then combined in all possible pairs, and an ordering of peptides for the vector is produced as follows:

To determine the optimal order of peptide–spacer–peptide combinations, binding predictions are made for all peptide registers overlapping the junction. A directed graph is then constructed, with nodes defined as target peptides, and edges representing junctions. The score of each edge is defined as the lowest binding score of its junctional peptides (a conservative metric). Edges with scores below the threshold are removed, and if heuristics indicate that a valid graph may exist, a simulated annealing procedure is used to identify a path through the nodes that maximizes junction scores (preserving the weakest overall predicted binding for junctional epitopes). If no valid ordering is found, additional "spacer" amino acids are added to each junction, binding affinities are recalculated, and a new graph is constructed and tested, setting edge weights equal to that of the best performing (highest binding score) peptide–junction–spacer combination.

The spacers used for pVACvector are set by the user with the --spacers parameter. This parameter defaults to None, AAY, HHHHH, GGSS, GPGPG, HLA, AAL, H, HHC, HHH, HH, HDHD, HHL, and HHHHC, where None is the placeholder for testing junctions without a spacer sequence. Spacers are tested iteratively, starting with the first spacer in the list and adding subsequent spacers if no valid path is found. If the user wishes to specify a custom spacer such as the furin cleavage site (e.g., R-X-(R/K)-R), they can do so (e.g., --spacers RKKK).

If no result is found after testing with the full set of spacers, the ends of “problematic” peptides, where all junctions contain at least one well-binding epitope, will be clipped by removing one amino acid at a time and then repeating the above binding and graph-building process. This clipping may be repeated up to the number of times specified in the --max-clip-length parameter.

It may be necessary to explore the parameter space when running pVACvector. As binding predictions for some sites vary substantially across algorithms, the most conservative settings may result in no valid paths, often due to one "outlier" prediction. Carefully selecting which predictors to run may help ameliorate this issue as well. In general, setting a higher binding threshold (e.g., 1,000 nmol/L) and using the median binding value (--top-score-metric median) will lead to greater possibility of a design, whereas more conservative settings of 500 nmol/L and lowest/best binding value (--top-score-metric lowest) will give more confidence that there are no junctional neoepitopes.

Our current recommendation is to run pVACvector several different ways and choose the path resulting from the most conservative set of parameters that produces a vector design containing all candidate neoantigens.

pVACapi and pVACviz

pVACapi is implemented using the Python libraries Flask and Boto3. The pVACviz client is written in TypeScript using the Angular Web application framework, the Clarity UI component library, and the ngrx library for managing application state.

Data availability

Data from 100 cases each of melanoma, hepatocellular carcinoma, and lung squamous cell carcinoma were obtained from TCGA (33) and downloaded via the GDC. This data can be accessed under dbGaP study accession phs000178. Data for demonstration and analysis of fusion neoantigens were downloaded from the GitHub repo for Integrate (https://github.com/ChrisMaherLab/INTEGRATE-Vis/tree/master/example).

Software availability

The pVACtools codebase is hosted publicly on GitHub at https://github.com/grifthlab/pVACtools and https://github.com/grifthlab/BGA-interface-projects (pVACviz). User documentation is available at http://www.pvactools.org. This project is licensed under the Non-Profit Open Software License version 3.0 (NPOSL-3.0, https://open source.org/licenses/NPOSL-3.0). pVACtools has been packaged and uploaded to PyPI under the "pvactools" package name and can be installed on Linux systems by running the "pip install pvactools" command. Installation requires a Python 3.5 environment which can be emulated by using Conda. Versioned Docker images are available on DockerHub (https://hub.docker.com/r/grifthlab/pvactools/). Releases are also made available on GitHub (https://github.com/grifthlab/pVACtools/releases).

Example pipeline for creation of pVACtools input files

pVACtools was designed to support a standard VCF variant file format and thus should be compatible with many existing variant calling pipelines. However, as a reference, we provide the following description of our current somatic and expression analysis pipeline (manuscript in preparation) which has been implemented using docker, common workflow language (CWL; ref. 34), and Cromwell (35). The pipeline consists of workflows for alignment of exome DNA and RNA sequencing (RNA-seq) data, somatic and germline variant detection, RNA-seq expression estimation, and HLA typing.

This pipeline begins with raw patient tumor exome and cDNA capture data (36) or RNA-seq data and leads to the production of annotated VCFs for neoantigen identification and prioritization with pVACtools.
Our pipeline consists of several modular components: DNA alignment, HLA typing, germline and somatic variant detection, variant annotation, and RNA-seq analysis. More specifically, we use BWA-MEM (15) for aligning the patient's tumor and normal exome data. For germline variant calling, the output BAM then undergoes merging (merging of separate instrument data; Samtools (37) Merge), query name sorting (Picard SortSam), duplicate marking (Picard MarkDuplicates), position sorting (Picard SortSam), and base quality recalibration (GATK BaseRecalibrator). GATK’s HaplotypeCaller (18) is used for germline variant calling, and the output variants were annotated using VEP (38) and filtered for coding sequence variants.

For somatic variant calling, the pipeline is consistent with above except that the variant calling step combines the output of four variant detection algorithms: Mutect2 (39), Strelka (40), Varscan (16, 17), and Pindel (41). The combined variants are normalized using GATK’s LeftAlignAndTrimVariants to left-align the indels and trim common bases. Vt (42) is used to split multiallelic variants. Several filters such as gnomAD allele frequency (maximum population allele frequency), percentage of mapq0 reads, as well as pass-only variants are applied prior to annotation of the VCF using VEP. We use a combination of custom and standard plugins for VEP annotation (parameters: --format VCF --plugin Downstream --plugin Wildtype --symbol --term SO --transcript_version --tsl --coding_only --flag_pick --hgvs).

Variant coverage is assessed using bam-readcount (https://github.com/genome/bam-readcount) for both the tumor and normal DNA exome data, and this information is also annotated into the VCF output using VTools (http://vtools.org).

Our pipeline also generates a phased-VCF file by combining both the somatic and germline variants and running the sorted combined variants through GATK ReadBackedPhasing.

For RNA-seq data, the pipeline first trims adapter sequences using Flexbar (43) and aligns the patient’s tumor RNA-seq data using HISAT2 (44). Two different methods, Stringtie (45) and Kallisto (46), are employed for evaluating both the transcript and gene expression values. In addition, coverage support for variants in RNA-seq data is determined with bam-readcount. This information is added to the VCF using VTools and serves as an input for neoantigen prioritization with pVACtools.

Optionally, our pipeline can also run HLA-typing in silico using OptiType (47) when clinical HLA typing is not available.

We aim to support various versions of these pipelines (via the use of standardized file formats) and are also actively developing an end-to-end public CWL workflow starting from sequence reads to pVACseq output (https://github.com/genome/analysis-workflows/blob/master/definitions/pipelines/immuno.cwl).

Results

The pVACtools workflow (Fig. 1) is divided into modular components that can be run independently. The main tools in the workflow...
are: (i) pVACseq: a significantly enhanced and reengineered version of our previous tool (48) for identifying and prioritizing neoantigens from a variety of tumor-specific alterations, (ii) pVACfuse: a tool for detecting neoantigens resulting from gene fusions, (iii) pVACviz: a graphical user interface Web client for process management, visualization, and selection of results from pVACseq, (iv) pVACvector: a tool for optimizing design of neoantigens and nucleotide spacers in a DNA vector that prevents high-affinity junctional neoantigens, and (v) pVACapi: an OpenAPI HTTP REST interface to the pVACtools suite.

pVACseq

pVACseq (48) has been reimplemented in Python3 and extended to include many new features since our initial report of its release. pVACseq no longer requires a custom input format for variants, and now uses a standard VCF file annotated with VEP (38). In our own neoantigen identification pipeline, this VCF is obtained by merging results from multiple somatic variant callers and RNA expression tools (as described in Materials and Methods: Example pipeline for creation of pVACTools input files). Information that is not natively available in the VCF output from somatic variant callers (such as coverage and VAFs for RNA and DNA, as well as gene and transcript expression values) can be added to the VCF using VATools (http://vatools.org), a suite of accessory routines that we created to accompany pVACtools. pVACseq queries these features directly from the VCF, enabling prioritization and filtering of neoantigens based on sequence coverage and expression information. In addition, pVACseq now makes use of phasing information taking into account variants proximal to somatic variants of interest. Because proximal variants can change the peptide sequence and affect neoantigen-binding predictions, this is important for ensuring that the selected neoantigens correctly represent the individual’s genome (11). We have also expanded the supported variant types for neoantigen predictions to include in-frame indels and frameshift variants. These capabilities expand the potential number of targetable neoantigens several-fold in many tumors (refs. 12, 14; Supplementary Fig. S1).

To prioritize neoantigens, pVACseq now offers support for eight different MHC Class I antigen prediction algorithms and four MHC Class II prediction algorithms. The tool does this in part by leveraging the IEDB (49) and their suite of six different MHC Class I prediction algorithms, as well as three MHC Class II algorithms (as described in Materials and Methods: Neoantigen prediction). pVACseq supports local installation of these tools for high-throughput users, access through a docker container (https://hub.docker.com/r/griffithlab/pvactools), and ready-to-go access via the IEDB RESTful Web interface. In addition, pVACseq now contains an extensible framework for supporting new neoantigen prediction algorithms that has been used to add support for two new non-IEDB algorithms—MHCflurry (26) and MHCnuggets (27). By creating a framework that integrates many tools, we allow for (i) a broader ensemble approach than IEDB and (ii) a system that other users can leverage to develop improved ensemble ranking, or to integrate proprietary or not-yet-public prediction software. This framework enables non-informatics expert users to predict neoantigens from sequence variant data sets.

Once neoantigens have been predicted, binding affinity values from the selected prediction algorithms are aggregated into a median binding score and a report file is generated. Finally, a rank is calculated that can be used to prioritize the predicted neoantigens. The rank takes into account gene expression, sequence read coverage, binding affinity predictions, and agreotopicity (as described in the Materials and Methods: Ranking of neoantigens). The pVACseq rank allows users to pick the exemplar peptide for each variant from a large number of lengths, registers, alternative transcripts, and HLA alleles. In addition to applying strict binding affinity cutoffs, the pipeline also offers support for MHC allele-specific cutoffs (50). The allele-specific binding filter incorporates allele-specific thresholds from IEDB for the 38 most common HLA-A and HLA-B alleles, representative of the nine major supertypes, instead of the 500 nmol/L default cutoff. Because these allele-specific thresholds only include cutoffs for HLA-A and HLA-B alleles, we recommend evaluating binding predictions for HLA-C alleles separately with a less stringent cutoff (such as 1,000 nmol/L) and merging results when picking final candidates.

We also offer cleavage position predictions via optional processing through NetChop (51) as well as stability predictions made by NetMHCSstabPan (52).

pVACfuse

Previous studies show that the novel protein sequences produced by known driver gene fusions frequently produce neoantigen candidates (53). pVACfuse provides support for predicting neoantigens from such gene fusions. One possible input to pVACfuse is a set of annotated fusion files from AG Fusion (54), thus enabling support for a variety of fusion callers such as STAR-Fusion (55), FusionCatcher (56), JAFFA (57), and TopHat-Fusion (58). pVACfuse also supports annotated BEDPE format from any fusion caller which can, for example, be generated using INTEGRATE-Neo (53). These variants are then assessed for presence of fusion neoantigens using predictions from any of the pVACseq-supported binding prediction algorithms.

pVACviz and pVACapi

Implementing cancer vaccines in a clinical setting requires multidisciplinary teams, which may not include informatics experts. To support this growing community of users, we developed pVACviz, which is a browser-based user interface that assists in launching, managing, reviewing, and visualizing the results of pVACtools processes. Instead of interacting with the tools via terminal/shell commands, the pVACviz client provides a modern Web-based user experience. Users complete a pVACseq (Fig. 2) process setup form that provides helpful documentation and suggests valid values for inputs. The client also provides views showing ongoing processes, their logs, and interim data files to aid in managing and troubleshooting. After a process has completed, users may examine the results as a filtered data table or as a scatterplot visualization—allowing them to curate results and save them as a CSV file for further analysis. Extensive documentation of the visualization interface can be found in the online documentation (https://pvactools.readthedocs.io/en/latest/pvacviz.html).

To support informatics groups that want to incorporate or build upon the pVACtools features, we developed pVACapi, which provides an HTTP REST interface to the pVACtools suite. It provides the API that pVACviz uses to interact with the pVACtools suite. Advanced users could develop their own user interfaces or use the API to control multiple pVACtools installations remotely over an HTTP network.

pVACvector

Once a list of neoantigen candidates has been prioritized and selected, the pVACvector utility can be used to aid in the construction of DNA-based cancer vaccines. The input is either the output file from pVACseq or a FASTA file containing peptide sequences.
pVACvector returns a neoantigen sequence ordering that minimizes the effects of junctional peptides (which may create novel antigens) between the sequences (Fig. 3). This is accomplished by using the core pVACseq module to predict the binding scores for each junctional peptide and by modifying junctions with spacer (59, 60) amino acid sequences, or by trimming the ends of the peptides in order to reduce reactivity. The final vaccine ordering is achieved through a simulated annealing procedure (61) that returns a near-optimal solution, when one exists (details can be found in the Materials and Methods: Implementation).

User community
pVACtools has been used to predict and prioritize neoantigens for several immunotherapy studies (62–64) and cancer vaccine clinical trials (e.g., NCT02348320, NCT03121677, NCT03122106, NCT03092453, and NCT03532217). We also have a large external

Figure 2.
pVACviz GUI client. pVACtools offers a browser-based graphical user interface, pVACviz, which provides an intuitive means to launch pipeline processes, monitor their execution, and analyze, export, or archive their results. To launch a process, users navigate to the Start Page (A) and complete a form containing all of the relevant inputs and settings for a pVACseq process. Each form field includes help text and provides typeahead completion where applicable. For instance, the alleles field provides a typeahead dropdown menu that filters available alleles to display only those alleles that match the text that the user has entered. Once a process is launched, a user may monitor its progress on the Manage Page (B), which lists all running, stopped, and completed processes. The Details Page (C) shows a process’s current log, attributes, and any results files as well as provides controls for stopping, restarting, exporting, and archiving the process. The results of pipeline processes may be analyzed on the Visualize Page (D), which displays a customizable scatterplot of a file’s rows. The x- and y-axes may be set to any column in the result set, and filters may be applied to values in any column. In addition, points may be selected on the scatter plot or data grid (not visible in this figure) for further analysis or export as CSV files.

Figure 3.
An example pVACvector output showing the optimum arrangement of candidate neoantigens for a DNA vector-based vaccine design. The figure depicts a circularized DNA insert carrying 10 encoded neoantigenic peptide sequences to be synthesized and encoded/decoded into a DNA plasmid. DNA sequences encoding each peptide are ordered (with use of spacer sequences where needed) to ensure there are no strong-binding junctional epitopes. Each neoantigenic peptide candidate is shown in blue, green, red, orange, purple, and brown. To minimize junctional epitope affinity, pVACvector adds spacer sequences where needed. These are depicted in black, along with the binding affinity value of the junctional epitope. Labels represent the gene name and amino acid change for each candidate.
user community that has been actively evaluating and using these packages for their neoantigen analysis and has aided in subsequent refinement of pVACtools through extensive feedback. The original “pvacseq” package has been downloaded over 48,000 times from PyPI, the “pvttools” package has been downloaded over 34,000 times, and the “pvttools” docker image has been pulled over 37,000 times.

Analysis of TCGA data using pVACtools

To demonstrate the utility and performance of the pVACtools package, we downloaded exome sequencing and RNA-seq data from TCGA (33) from 100 cases each of melanoma, hepatocellular carcinoma, and lung squamous cell carcinoma and used patient-specific MHC Class I alleles (Supplementary Fig. S2) to determine neoantigen candidates for each patient. There were a total of 64,422 VEP-annotated variants reported across 300 samples, with an average of 214 variants per sample. Of these, 61,486 were single nucleotide variants (SNV), 479 were in-frame insertions and deletions, and 2,465 were frameshift mutations (Supplementary Fig. S1). We used this annotated list of variants as input to the pvacseq component of pVACtools to predict neoantigenic peptides. pvacseq reported 14,599,993 unfiltered neoantigen candidates. The original version of pvacseq (20) reported 10,284,467 neoantigens. This demonstrated that, by extending support for additional variant types as well as prediction algorithms (due to support for additional alleles), we produced 42% more raw candidate neoantigens.

After applying our default median binding affinity cutoff of 500 nmol/L across all eight MHC Class I prediction algorithms, there were 96,235 predicted strong-binding neoantigens, derived from 34,552 somatic variants (32,788 missense SNVs, 1,603 frameshift variants, and 131 in-frame indels). This set of strong binders was further reduced by filtering out mutant peptides with median predicted binding affinities (across all prediction algorithms) greater than that of the corresponding wild-type peptide (i.e., mutant/wild-type binding affinity fold change >1), resulting in 70,628 neoantigens from 28,588 variants (26,880 SNVs, 1,583 frameshift, and 125 in-frame indels).

This set was subsequently filtered by evaluation of exome sequencing data coverage and our recommended defaults as follows: by applying the default criteria of VAF cutoff of >25% in tumors and <2% in normal samples, with at least 10× tumor coverage and at least 5× normal coverage, 10,730 neoantigens from 4,891 associated variants (4,826 SNVs, 56 frameshift, and 9 in-frame indels) were obtained, with an average of 36 neoantigens predicted per case. Because RNA-seq data also were available, the filtering criteria included RNA-based coverage filters (tumor RNA VAF >25% and tumor RNA coverage >10×) as well as a gene expression filter (FPKM >1). To condense the results even further, only the top ranked neoantigen was selected per variant (“top-score filter”) across all alleles, lengths, and registers (position of amino acid mutation within peptide sequence), resulting in 4,891 total neoantigens with an average of 16 neoantigens per case. This list was then processed with pVACvector to determine the optimum arrangement of the predicted high-quality neoantigens for a DNA vector–based vaccine design (Fig. 3).

Correlations and bias among binding prediction tools

Because we offer support for as many as eight different Class I binding prediction tools, we assessed agreement in binding affinity predictions (IC50) between these algorithms from a random subset of 100,000 neoantigens peptides from our TCGA analysis, described above (Fig. 4). The highest correlation was observed between the two algorithms that are based on a stabilization matrix method (SMM) —SMM and SMMPMBEC. The next best correlation was observed between NetMHC and MHCflurry, possibly due to both being allele-specific predictors employing neural network-based models. Overall, the correlation between prediction algorithms is low (mean correlation of 0.388 and range of 0.18–0.89 between all pairwise comparisons of algorithms).

We also evaluated if there were any biases among the algorithms to predict strong-binding epitopes (i.e., binding affinity ≤ 500 nmol/L) by plotting the number of predicted peptides based on their respective strong-predicting algorithms (Fig. 5; Supplementary Fig. S3). We found that MHCnuggets (v2.2) predicted the highest number of strong-binding candidates alone (Fig. 5A). Of the total number of strong-binding candidates predicted, 64.7% of these candidates were predicted by a single algorithm (any one of the eight algorithms), 35.2% were predicted as strong-binders by two to seven algorithms, and only 1.8% of the strong-binding candidates were predicted as strong binders by the combination of all eight algorithms (Supplementary Fig. S3). In fact, as shown in Fig. 6, even if one (or more) algorithms predict a peptide to be a strong binder, often another algorithm will disagree by a large margin, in some cases predicting that same peptide as a very weak binder. This remarkable lack of agreement underscores the potential value of an ensemble approach that considers multiple algorithms.

Next we determined if the number of human HLA alleles supported by these eight algorithms differed. As shown (Supplementary Fig. S4), MHCnuggets supports the highest number of human HLA alleles.

Comparison of filtering criteria

Because pVACtools offers a multitude of ways to filter a list of predicted neoantigens, we evaluated and compared the effect of each filter for selecting high-quality neoantigen candidates. There were 14,599,993 unfiltered neoantigen candidates predicted by pvacseq resulting from 64,422 VEP-annotated variants reported across 300 samples.

We first compared the effect of running pVACtools with the commonly used standard binding score cutoff of ≤ 500 nmol/L (parameters: -b 500) to the newly added allele-specific score filter (parameters: -a). These cutoffs were, by default, applied to the
"median" binding score of all prediction algorithms. In total, 96,235 neoantigens (average 320.8 per patient) were predicted using the 500 nmol/L binding score cutoff compared with 94,068 neoantigens (average 313.6 per patient) using allele-specific filters. About 79% of neoantigens were shared between the two sets.

We also further narrowed these sets to include only those predictions where the default "median" predicted binding affinities (≤500 nmol/L) were lower than each corresponding median wild-type peptide affinity (parameters: -b 500 -c 1; i.e., a binding affinity mutant/wild-type ratio or differential agretopicity value indicating that the mutant version of the peptide is a stronger binder; ref. 21). Using the agretopicity value filter, 70,628 neoantigens (average 235.43 per patient) were predicted versus the previously reported 96,235 neoantigens without this filter.
We also evaluated the effect of the agretopicity value filter when applied to the set of peptides filtered on "lowest" binding score of ≤500 nmol/L (parameters: -b 500 -c 1 -m lowest). This filter limits peptides to those where at least one of the algorithms predicts a strong binder, instead of calculating a median score and requiring that to meet the 500 nmol/L threshold (Fig. 6). Using the lowest binding score filter resulted in an 11-fold increase in the number of candidates predicted (827,423 candidates, average 2,758.08 per patient). This approach may be useful for finding candidates in tumors with low mutation burden (but may also presumably lead to a higher rate of false positives).

Using the median (default) binding affinity filtering criteria, we next applied coverage and expression-based filters. First, we filtered using the recommended defaults, i.e., greater than 5× normal DNA coverage, less than 2% normal VAF, greater than 10× tumor RNA and DNA coverage, and greater than 25% tumor RNA and DNA VAF, along with FPKM >1 for transcript level expression (parameters: --normal-cov 5 --tdna-cov 10 --trna-cov 10 --normal-vaf 0.02 --tdna-vaf 0.25 --trna-vaf 0.25 --expn-val 1). A total of 10,730 neoantigens were shortlisted across all samples with an average of 35 neoantigens per case. We then compared this set with slightly more stringent criteria using tumor DNA and RNA VAF of 40% (parameters: --tdna-vaf 0.40 --trna-vaf 0.40). This shortened our list of predicted neoantigens to 4,073 candidates with an average of 13 candidates per patient.

Demonstration of neoantigen analysis using pVACfuse

To demonstrate the potential of neoantigens resulting from gene fusions, we analyzed TCGA prostate cancer RNA-seq data from 302 patients. This dataset was previously used as a demonstration set for fusion neoantigen prediction supported by INTEGRATE-Neo (22). We wanted to assess the difference (if any) in neoantigen candidates report by INTEGRATE-Neo using the one MHC Class I prediction algorithm it supports (NetMHC) versus an ensemble of eight Class I prediction algorithms supported in pVACfuse.

Using 1,619 gene fusions across 302 samples as input, pVACfuse reported 2,104 strong-binding neoantigens (binding affinity ≤500 nmol/L) resulting from 739 gene fusions. On average, there were about seven neoantigens per sample resulting from an average of two fusions per case. This is an 8-fold increase in the number of strong-binding neoantigens predicted by pVACfuse versus those reported by INTEGRATE-Neo alone, which reported 261 neoantigens across 210 fusions.

Validation of pVACtools results

To assess the real-world validity of pVACtools results, we reanalyzed raw sequencing data from published studies that had performed immunologic validation of candidate neoantigens (Supplementary Tables S2 and S3). Analysis of a patient with melanoma (pt3713; ref. 65) showed that pVACtools was able to recapitulate eight (80%) of the positively validated peptides in the filtered list of neoantigens using our suggested filtering methodology (median binding predicted IC50 <500 nmol/L, ≥5× DNA and RNA coverage, >10% DNA and RNA VAF, >1 TPM gene expression). Two of the positively validated peptides were filtered out due to low RNA depth and VAF values but had median binding affinity predictions of 13.707 and 4.799 nmol/L. We were also able to eliminate 68% of negatively validated peptides using these filters. In an additional analysis of a patient with chronic lymphocytic leukemia (pt5002; ref. 66), we were able to recapitulate two (100%) of the positively validated peptides in our filtered list of neoantigens using our suggested filtering methodology (median binding predicted IC50 <500 nmol/L, ≥5× DNA coverage, >10% DNA.
tumor sequencing analysis pipeline and reports a VAF). No RNA data were available for this sample. We were also able to eliminate 57% of negatively validated peptides using these filters.

**Discussion**

As reported from our demonstration analysis, a typical tumor has too many possible neoantigen candidates to be practical for a vaccine. There is therefore a critical need for a tool that takes in the input from a tumor sequencing analysis pipeline and reports a filtered and prioritized list of neoantigens. pVACTools enables a streamlined, accurate, and user-friendly analysis of neoantigenic peptides from NGS cancer datasets. This suite offers a complete and easily configurable end-to-end analysis, starting from somatic variants and gene fusions (pVACseq and pVACfuse respectively), through filtering, prioritization, and visualization of candidates (pVACviz), and determining the best arrangement of candidates for a DNA vaccine vector (pVACvector). By supporting additional classes of variants as well as gene fusions, we predict an increased number of predicted neoantigens which may be critical for low mutational burden tumors. Finally, by extending support for multiple binding prediction algorithms, we allow for a consensus approach. The need for this integrated approach is made abundantly clear by the disagreement between the candidate neoantigens identified by these algorithms, as observed in our demonstration analyses.

To support a wider range of variant events not easily representable in VCF format such as alternatively spliced transcripts that may create tumor antigens (67), upcoming versions of pVACTools will add a new tool, pVACbind, which can be used to execute our prediction pipeline from sequences contained in a FASTA input file. Another major planned feature is the addition of reference proteome similarity assessment of predicted peptides in order to exclude peptides that are not true neoantigens because they occur in other parts of the proteome. Upcoming versions will also include the calculation of various metrics for peptide manufacturability for use in synthetic long peptide-based vaccines. In addition, we are planning on adding binding affinity percentile ranks to our prediction outputs for those prediction algorithms that support it. This would also allow us to support additional IEDB prediction algorithms, such as Combinatorial Library (68) and Sturniolo (69). The IEDB software added support for various Class II epitope lengths starting with version 2.22. We plan on updating pVACTools in an upcoming version to take advantage of this new functionality. Lastly, we are working on a CWL immunotherapy pipeline that will execute all steps starting with alignment, somatic and germline variant calling, HLA-typing, and RNA-seq analysis, including fusion calling, followed by running pVACseq and pVACfuse. This will support users that would like a full end-to-end solution not supported by pVACTools itself.

The results from pVACTools analyses are already being used in dozens of cancer immunology studies, including studying the relationship between tumor mutation burden and neoantigen load to predict response to immunotherapies (64, 70, 71), and the design of cancer vaccines in ongoing clinical trials (e.g., NCT02348320, NCT03121677, NCT03122106, NCT03092453, and NCT03535217). Whether the predicted neoantigens will induce T-cell responses remains an open question for the field (72–74). Pipelines like pVACtools are meant to assist these groups to obtain a prioritized set of neoantigen candidates from among a myriad of possible epitope predictions. Pursuant to that, they should perform additional functional validations according to the standards in this field (73). Whereas it may not be possible to evaluate a de novo response to all predicted candidate neoantigens prior to vaccination, we highly recommend adding a similar validation step as part of the clinical workflow to determine those candidates that have a preexisting immune response. Validation data from the studies currently using pVACtools will be incorporated into our toolkit as they become available, and we hope that over time this will increase the prioritization process. We anticipate that pVACtools will make such analyses more robust, reproducible, and facile as these efforts continue.

**Disclosure of Potential Conflicts of Interest**

No potential conflicts of interest were disclosed.

**Authors’ Contributions**

**Conception and design:** J. Hundal, S. Kiwala, J. McMichael, W.E. Gillanders, E.R. Mardis, O.L. Griffith, M. Griffin


**Acquisition of data (provided animals, acquired and managed patients, provided facilities, etc.):** J. Hundal, C.J. Liu

**Analysis and interpretation of data (e.g., statistical analysis, biostatistics, computational analysis):** J. Hundal, H. Xia, C.J. Liu, S. Zhao, M. Griffith

**Writing, review, and/or revision of the manuscript:** J. Hundal, S. Kiwala, C.A. Miller, H. Xia, C.J. Liu, W.E. Gillanders, E.R. Mardis, O.L. Griffith, M. Griffin

**Administrative, technical, or material support (i.e., reporting or organizing data, constructing databases):** J. Hundal, S. Kiwala, J. McMichael, A.T. Wollam, Y.-Y. Feng, A.Z. Wollam, M. Neveau, J. Walker

**Study supervision:** J. Walker, E.R. Mardis, O.L. Griffith, M. Griffin

**Acknowledgments**

We thank the patients and their families for donation of their samples and participation in clinical trials. We also thank our growing user community for testing the software and providing useful input, critical bug reports, as well as suggestions for improvement and new features. We are grateful to Drs. Robert Schreiber, Gavin Dunn, and Beatriz Carreno for their expertise and guidance on foundational work on cancer immunology using neoantigens and suggestions on improving the software. O.L. Griffith was supported by the NCI of the NIH under award numbers U01CA209936, U01CA231844, and U24CA237719. M. Griffin was supported by the National Human Genome Research Institute (NHGRI) of the NIH under award number U01HG007940 and the V Foundation for Cancer Research under award number V2018-007.

The costs of publication of this article were defrayed in part by the payment of page charges. This article must therefore be hereby marked advertisement in accordance with 18 U.S.C. Section 1734 solely to indicate this fact.

Received May 29, 2019; revised October 6, 2019; accepted December 30, 2019; published first online January 6, 2020.

**References**


pVACtools: Toolkit for Tumor Neoantigen Characterization


Cancer Immunology Research

pVACtools: A Computational Toolkit to Identify and Visualize Cancer Neoantigens


Updated version
Access the most recent version of this article at:
doi:10.1158/2326-6066.CIR-19-0401

Supplementary Material
Access the most recent supplemental material at:
http://cancerimmunolres.aacrjournals.org/content/suppl/2020/01/04/2326-6066.CIR-19-0401.DC1

Cited articles
This article cites 64 articles, 11 of which you can access for free at:
http://cancerimmunolres.aacrjournals.org/content/8/3/409.full#ref-list-1

Citing articles
This article has been cited by 14 HighWire-hosted articles. Access the articles at:
http://cancerimmunolres.aacrjournals.org/content/8/3/409.full#related-urls

E-mail alerts
Sign up to receive free email-alerts related to this article or journal.

Reprints and Subscriptions
To order reprints of this article or to subscribe to the journal, contact the AACR Publications Department at pubs@aacr.org.

Permissions
To request permission to re-use all or part of this article, use this link
http://cancerimmunolres.aacrjournals.org/content/8/3/409.
Click on "Request Permissions" which will take you to the Copyright Clearance Center's (CCC) Rightslink site.