Semi-Supervised Learning for Somatic Variant Calling and Peptide Identification in Personalized Cancer Immunotherapy

Elham Sherafat[1], Jordan Force[1], and Ion I.Măndoiu[1]

[1]University of Connecticut, Computer Science and Engineering Department, 371 Fairfield Way, Unit 4155, Storrs, CT 06269-4155, E-mail: {elham.sherafat,jordan.force,ion.mandoiu}@uconn.edu

Abstract

Personalized cancer vaccines are emerging as one of the most promising approaches to immunotherapy of advanced cancers. However, only a small proportion of the neoepitopes generated by somatic DNA mutations in cancer cells lead to tumor rejection. Since it is impractical to experimentally assess all candidate neoepitopes prior to vaccination, developing accurate methods for predicting *tumor-rejection mediating neoepitopes* (TRMNs) is critical for enabling routine clinical use of cancer vaccines. In this paper we introduce *Positive-unlabeled Learning using AuTOml* (PLATO), a general semi-supervised approach to improving accuracy of model-based classifiers. PLATO generates a set of high confidence positive calls by applying a stringent filter to model-based predictions, then rescores remaining candidates by using positive-unlabeled learning. To achieve robust performance on clinical samples with large patient-to-patient variation, PLATO further integrates AutoML hyper-parameter tuning, classification threshold selection based on spies, and support for bootstrapping. Experimental results on real datasets demonstrate that PLATO has improved performance compared to model-based approaches for two key steps in TRMN prediction, namely somatic variant calling from exome sequencing data and peptide identification from MS/MS data.

1 Background

Personalized cancer vaccines are emerging as a promising alternative to nonspecific treatments such as chemotherapy in the management of advanced cancers [1, 2]. This approach harnesses the power of the patient's own immune system to attack cells that express immunogenic peptides called *neoepitopes*. Neoepitopes are generated as a result of somatic DNA mutations that arise in cancer cells, hence making the immune response tumor-specific. However, only a small proportion of the potential neoepitopes lead to tumor rejection [3, 4, 5, 6]. Methods for predicting *tumor-rejection mediating neoepitopes* (TRMNs) are the subject of much active research, including large consortium efforts such as the Tumor Neoantigen Selection Alliance [7].

Existing bioinformatics pipelines for neoepitope and TRMN prediction (e.g., [8, 9, 10, 11]) include two main steps: (1) calling tumor-specific somatic variants from matched tumor-normal exome or whole-genome sequencing data, and (2) predicting which mutated peptides generated by non-synonymous somatic variants are presented to the immune system by the Major Histocompatibility Complex (MHC) alleles of the patient. Recent experimental work using a mouse tumor model [12] supports the utility of incorporating a third step, which prioritizes for vaccination the mutated peptides detected by tandem mass-spectrometry (MS/MS) in elutions of peptide-MHC complexes recovered from the surface of tumor cells.

Although many bioinformatics tools exist for each of these steps, there is still significant room for improvement. In particular, although many somatic variant callers have been developed based on diverse statistical models, agreement between them remains low [13, 14]. Key impediments to achieving consistently high accuracy with model-based methods include the large patient-to-patient variation in tumor purity and heterogeneity, sequencing library preparation artifacts, sequencing errors, and data processing errors such as incorrect read alignment. Several machine learning methods for somatic mutation calling have been recently

Table 1: Variant calling performance on sequencing datasets P1-P4 generated for four ovarian cancer patients.

DataSet	Caller	#calls	TP	FP	TN	FN	TPR	PPV	F1
P1	SNVQ	515	48	114	0	0	100	29.63	45.71
P: 65	Strelka	435	48	34	80	0	100	58.54	73.85
U: 29758	2CP	65	41	8	106	7	85.42	83.67	84.54
Reseq: 162	PLATO	587	42	6	104	10	80.77	87.5	84
P2	SNVQ	597	147	65	3	2	98.66	69.34	81.44
P: 187	Strelka	619	149	59	9	0	100	71.63	83.47
U: 31210	2CP	187	133	39	29	16	89.26	77.33	82.87
Reseq: 217	PLATO	449	144	43	25	5	96.64	77.01	85.71
P3	SNVQ	629	61	13	8	1	98.39	82.43	89.71
P: 76	Strelka	306	62	11	10	0	100	84.93	91.85
U: 30289	2CP	76	57	1	20	5	91.94	98.28	95
Reseq: 83	PLATO	429	62	3	18	0	100	95.38	97.64
P4	SNVQ	482	48	23	87	2	96	67.61	79.34
P: 67	Strelka	380	50	94	16	0	100	34.72	51.55
U: 30176	2CP	67	45	2	108	5	90	95.74	92.78
Reseq: 160	PLATO	490	48	7	103	2	96	87.27	91.43

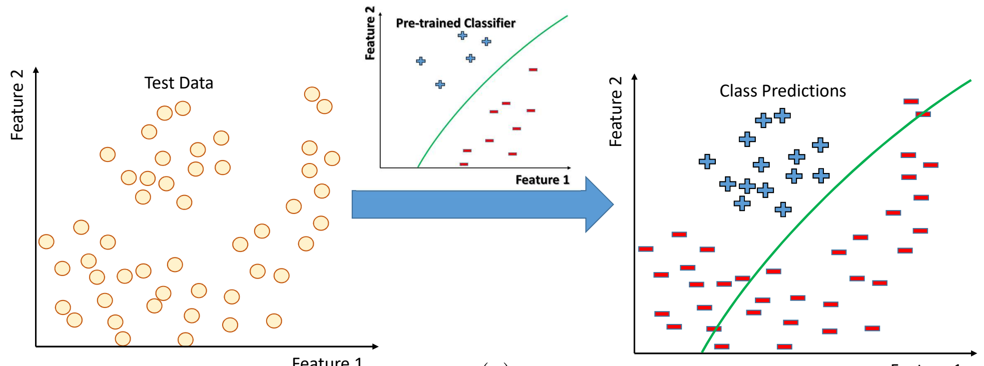
developed to address this challenge [15, 16, 17, 18, 19, 20, 21, 22]. However, most of these methods adopt a supervised learning paradigm and generally require large amounts of training data.

In this paper we introduce a novel machine learning approach aimed at increasing the sensitivity of any existing model-based pipeline for somatic variant calling while maintaining a high positive predictive value. To achieve robust performance despite the significant patient-to-patient variation present in clinical samples, we adopt a semi-supervised approach that learns salient attributes from the data itself, without a need for prior training datasets. Our approach, referred to as *Positive-unlabeled Learning using AuTOml* (PLATO), is illustrated in Figure 1 (see also the flowchart in Figure 2). PLATO takes as input the list of unfiltered candidate somatic variant calls generated using an existing model-based pipeline along with a subset of highly confident calls obtained by applying stringent thresholds. PLATO adopts a *Positive-Unlabeled* (PU) learning approach, in which the set of highly confident calls are used as positive examples and the remaining candidate calls are used as unlabeled examples. Real cancer datasets have typical unlabeled:positive ratios of 1000:1 or higher. The vast majority of unlabeled examples are *a priori* expected to be true negatives (sequencing errors or germline variants). PLATO takes advantage of this skewed distribution to generate likely negative datasets by informed undersampling, i.e., randomly picking points that are furthest from the positive set according to the Gower distance in a space defined by categorical and numerical features such as confidence scores and allele coverage information generated by the model-based pipeline and sequence properties extracted from the genome and alignment files. PLATO then trains a classifier to discriminate between the positive and likely negative examples, and uses this classifier to label remaining data points. Hyper-parameter tuning is performed by cross-validation using the AutoML service provided by Microsoft Azure. Additionally, PLATO uses a "spy" approach for robust classification threshold selection, and performs a user specified number of bootstraps, reporting only variants with 50% or higher bootstrap support.

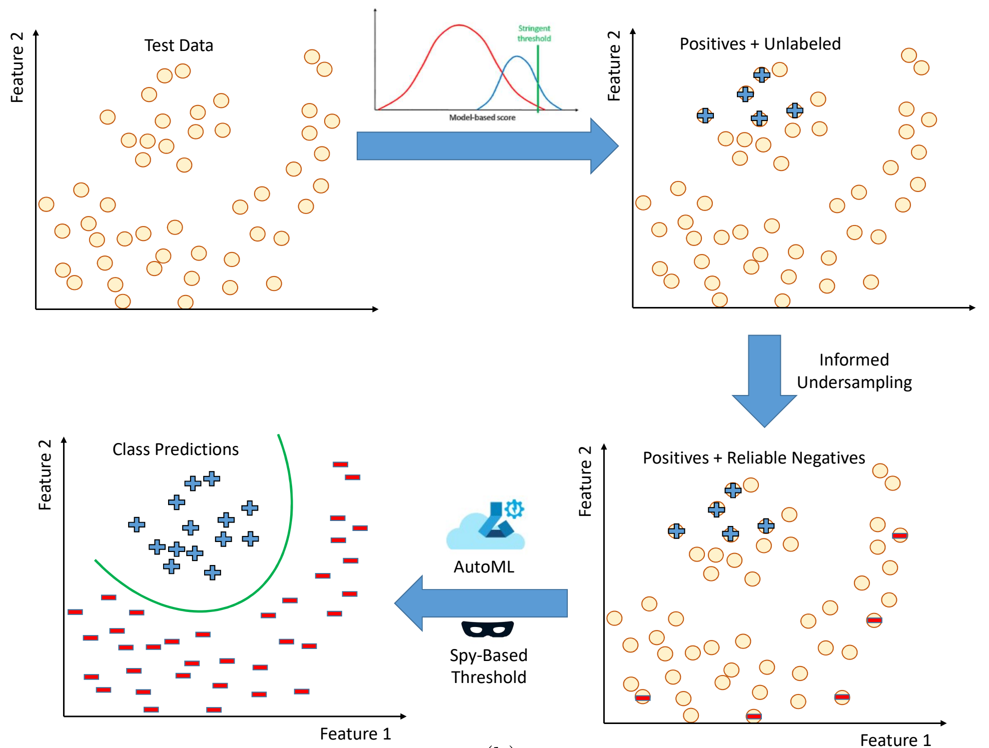
2 Results

2.1 Somatic variant calling from multi-technology exome sequencing data

To assess PLATO's accuracy we used matched normal-tumor exome sequencing data generated for four ovarian cancer patients (identified in this article as P1 to P4) using two different sequencing technologies, Illumina and Ion Torrent. The unlabeled set given as input to PLATO was generated using the *Consensus Caller Cross-Platform* (CCCP) Galaxy tool available as part of the GeNeo immunogenomics toolbox [23]. CCCP incorporates two state-of-the-art somatic mutation callers, SNVQ [24] and Strelka [25], and has the



(a)



(b)

Figure 1: Schematic representation of supervised classification (a) vs. PLATO's PU learning approach (b). Supervised classification requires training data and can perform poorly when the distributions of training and test data do not match. PU learning uses an existing model-based classifier with stringent thresholds and informed undersampling to train a classifier from the data itself.

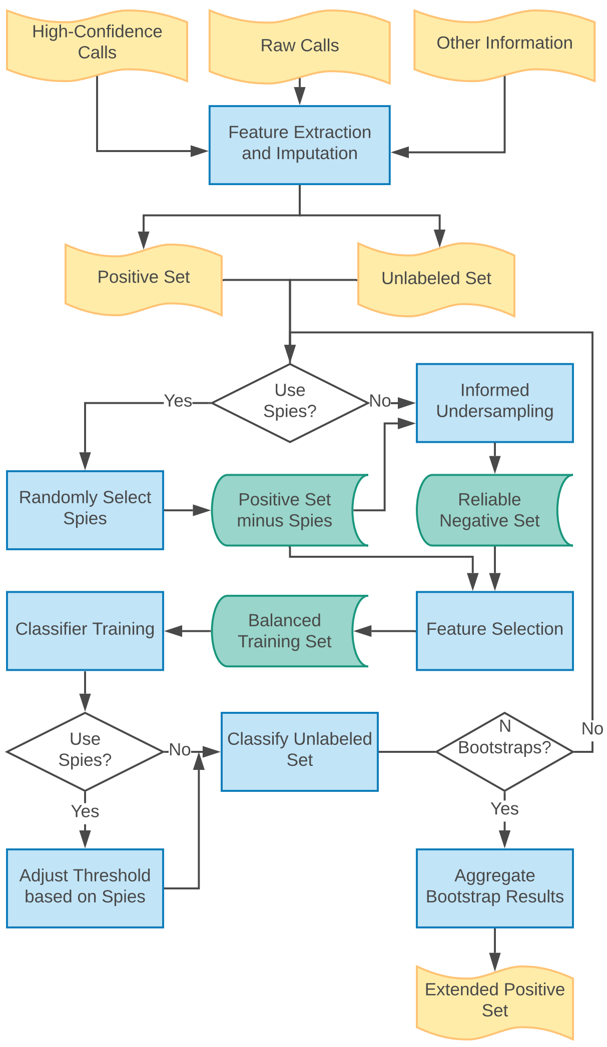
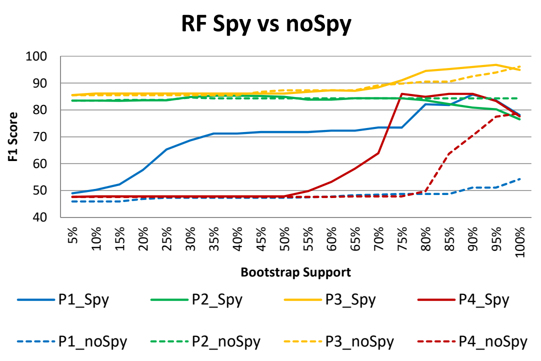
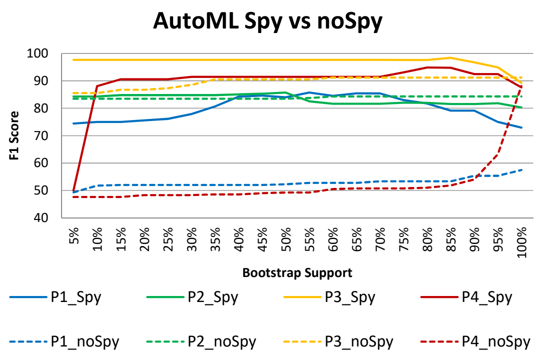


Figure 2: PLATO flowchart.

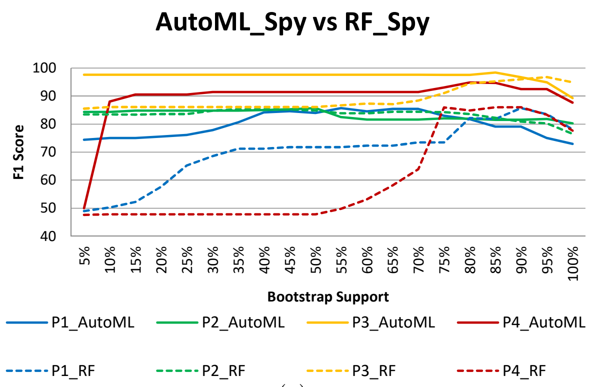
ability to process multi-technology sequencing data. Positive calls were generated by applying the 2CP filter [26] on the raw output of CCCP. 2CP requires that at least one of the two callers make a high confidence call from each of the two sequencing technologies. The only exception is when one of the sequencing technologies yields no read coverage, in which case both callers must make confident calls from the reads generated by the complementary technology. The ground truth for a subset of the predicted somatic variants was established by taking the consensus of calls made from high-depth targeted re-sequencing of amplicons generated using the AccessArray system from three or more replicates per patient of both tumor and normal tissue. The first column of Table 1 gives the number of resequenced variants for each patient along with the sizes of the P and U sets. In all cases, the resequenced set included all variants that passed the 2CP filter and for which AccessArray primers could be successfully designed using the primer design tool in GeNeo. The resequenced sets also included additional SNVs called using a random forest classifier at varying levels of bootstrap support. For each compared method we computed the number of *true positives (TP)*, *false positives (FP)*, *true negatives (TN)*, and *false negatives (FN)* relative to the set of variant calls for which the ground truth was available. The reported *true positive rate*, $TPR := TP/(TP + FN)$, *positive predictive value*, $PPV := TP/(TP + FP)$, and *F1 score*, $F1 := 2 \cdot TPR \cdot PPV/(TPR + PPV)$, were also computed relative to the ground truth available for each method.



(a)



(b)



(c)

Figure 3: F1 scores obtained by running PLATO with $N = 20$ bootstraps. (a) Random forest classification with spies-based classification threshold vs. 0.5 default, (b) AutoML classification with spies vs. 0.5 default, and (c) AutoML with spies vs. random forest with spies. P1-P4 denote the sequencing datasets generated for four different ovarian cancer patients.

2.1.1 Effect of classification threshold selection and classification algorithm

The users of PLATO can choose between automatic classification threshold selection based on spies or using the underlying classifier's default threshold (typically 0.5). Also, in principle, the PLATO framework can be used in conjunction with any supervised classification algorithm. AutoML already integrates a wide range of supervised classification methods, dynamically evaluating them on each dataset using a cross-validation approach to avoid over-fitting. However, using AutoML does come with an added computational cost. To see if this added cost is warranted, we compared the AutoML-based implementation of PLATO with a baseline implementation based on random forests.

Figures 3(a)-(b) show that, for both the random forest and AutoML implementations, using spies-based classification thresholds yields F1 scores close to and often better than those obtained by using the classifier's default threshold. This holds independently of the bootstrap support required for positive classification. Furthermore, Figure 3(c) shows that, when using spies-based thresholds, the AutoML-based implementation of PLATO has F1 score comparable to or better than those of the random forest implementation at virtually all bootstrap support cutoffs.

2.1.2 Comparison with model-based callers

Table 1 gives detailed accuracy results on the four ovarian cancer datasets, comparing PLATO with model-based callers SNVQ [24] and Strelka [25], as well as the 2CP filter of CCCP [26]. PLATO results in this table were obtained by using AutoML as classifier, spies-based classification threshold selection, $N = 20$ bootstraps, and 50% bootstrap support. On all four datasets, the F1 score of PLATO is comparable to or

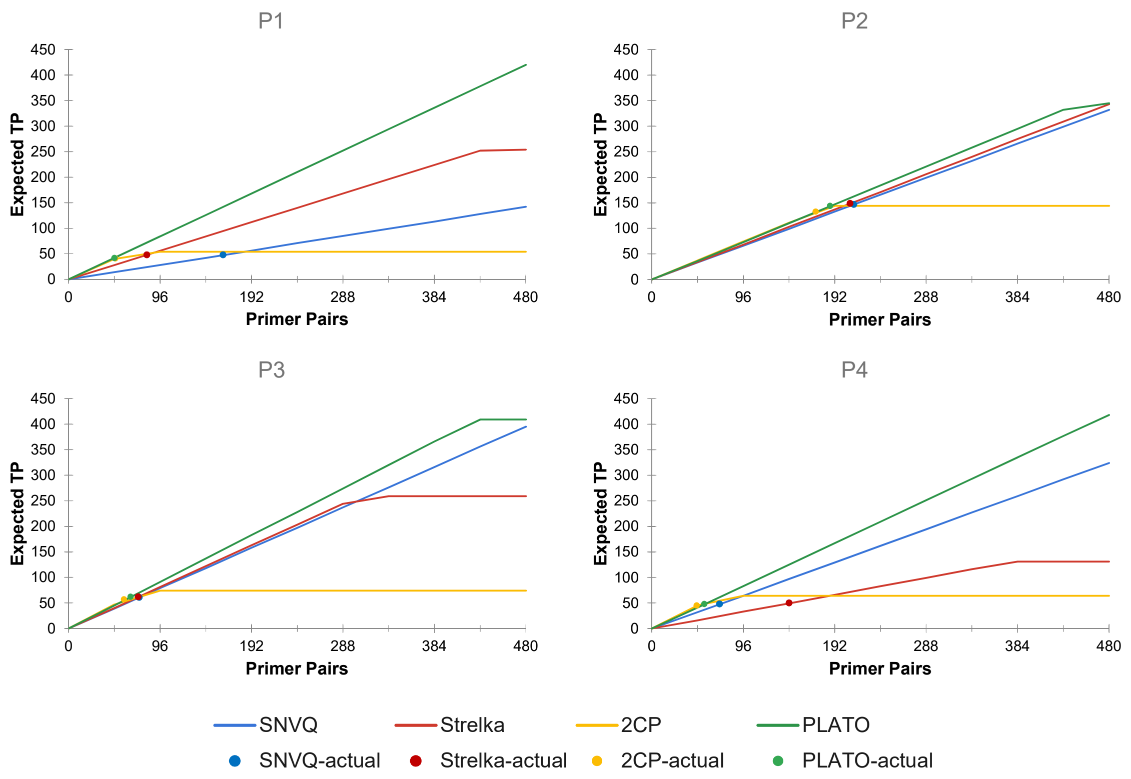
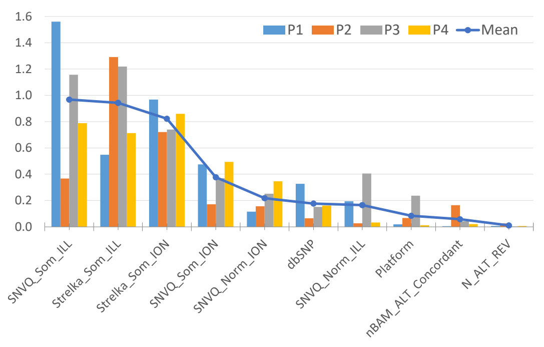


Figure 4: Expected TP count at different multiplexing rates for SNVQ, Strelka, 2CP, and PLATO run using AutoML, spies-based classification threshold selection, and 50% bootstrap support. The dots represent TP counts from the actual AccessArray resequencing experiment reported in Table 1. P1-P4 denote the sequencing datasets generated for four different ovarian cancer patients.

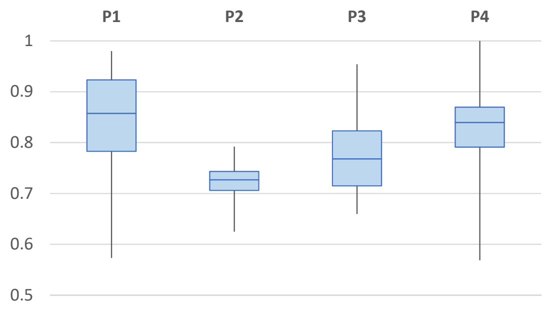
better than that of 2CP, which in turn is comparable to or better than that of SNVQ and Strelka. Unlike SNVQ and Strelka, PLATO always retains a high PPV, comparable to or better than that of 2CP. This is important, since PLATO also makes between 2.4× and 9× more calls than the very stringent 2CP filter. Assuming a constant PPV this suggests that substantially more SNVs are expected to be confirmed when resequencing candidates called by PLATO on the AccessArray. Figure 4 shows for each caller the expected TP count assuming a constant PPV for up to 480 primer pairs multiplexed on a 48.48 AccessArray IFC. For reference, Figure 4 also includes dots representing the counts from the actual AccessArray resequencing experiment reported in Table 1.

2.1.3 Feature importance for SNV calling

Figure 5(a) gives the importance reported by AutoML for the top 10 features used for SNV calling, averaged for each dataset over 20 bootstrap runs (for feature descriptions see the appendix). Not surprisingly, the top four features are the binary somatic calls made for each sequencing technology (Illumina and Ion Torrent) by the two callers integrated in CCCP (SNVQ and Strelka). Binary calls made by SNVQ from the normal Illumina and Ion Torrent exomes and dbSNP status follow close behind in importance. The variation in feature importance from dataset to dataset is remarkably high, underscoring the need for semi-supervised methods such as PLATO that can adapt to the idiosyncrasies of each dataset. Figure 5(b) gives boxplots of the classification cutoffs selected using the spy approach over the 20 bootstraps runs performed for each of the four datasets. Most likely due to the over-representation of negatives in the list of CCCP candidates, the spy-based cutoffs are always higher than the 0.5 default. Furthermore, the cutoff distributions vary from patient to patient, again underscoring PLATO's ability to adapt to each dataset.



(a)



(b)

Figure 5: Average feature importance for SNV calling (a), and boxplots of the classification cutoffs selected using the spy approach (b) over the 20 bootstraps runs performed for the P1-P4 ovarian cancer datasets.

2.2 Peptide identification from MS/MS data

For peptide identification, we evaluated our method on twenty datasets generated by [27] and retrieved from the ProteomeXchange repository using project identifier PXD004894. We retrieved the RAW MS/MS files for five different melanoma patients (identified as mel3, mel4, mel5, mel8, and mel12). For each patient we retrieved four MS/MS files, corresponding to two biological replicates per patient (p1/p2) and two independent MS/MS runs per replicate (identified by the date of the run, 2014-03-04/2014-03-05 or 2014-03-06). Table 2 gives the number of peptides identified at a q-value cutoff of 0.01 by MS-GF+, Percolator, and PLATO. Although both Percolator and MS-GF+ can compute PSM and Peptide level q-values, the q-values for all three methods were computed by our implementation of the procedure described in the Methods section to ensure that differences in peptide counts between the different methods are not due to variations in the q-value computation method.

For comparison, Table 2 also includes the number of peptides identified in [27] using the MaxQuant search engine with the same FDR cutoff. While we provide these numbers as a baseline, they should be considered with caution, because MaxQuant was used to search a different human proteome database. For MS-GF+ searches we used 20,585 protein sequences retrieved from Uniprot in 2019 (see the appendix for details), while [27] searched a database containing 85,919 protein sequences retrieved in 2014. As shown in Table 2 and visualized as improvement over the MaxQuant baseline in Figure 6, both PLATO and Percolator significantly outperform MaxQuant and MS-GF+ in terms of the number of peptides identified at 1% peptide-level FDR. Although their perfomance is comparable, PLATO has a slight edge over Percolator, outperforming it on 15 out of the 20 datasets while being outperformed only 5 times.

2.2.1 Feature importance for peptide identification

The top 15 features ranked by average AutoML importance are shown in Figure 7 (for feature descriptions see the appendix). The importance score of the lnEValue dominates the scores of the other features by more than one order of magnitude, and has relatively small sample-to-sample variation. Interestingly, the amino acids at known anchor positions for MHC class I binding have relatively low importance scores, most likely due to the fact that clinical MS/MS samples are comprised of peptides presented by up to six distinct MHC class I alleles, each with potentially different anchor position specificities.

3 Discussion

Experimental validation results on sequencing data from four ovarian cancer patients demonstrate the effectiveness of PLATO when combined with the existing *Consensus Caller Cross-Platform* (CCCP) pipeline for somatic variant calling [23]. Since the PU learning framework is broadly applicable, we also applied PLATO to improve the rate of confident peptide identification from tandem mass-spectrometry data. Specifically, we

Table 2: Number of peptides identified at 1% FDR from 20 MS/MS datasets generated by [27]. MHC-bound peptides were eluted from melanoma samples collected from five different patients (identified as mel3, mel4, mel5, mel8, and mel12), with two biological replicates (p1/p2) per patient, each analyzed on two independent MS/MS runs (identified by the date in the sample ID). For each MS/MS dataset, the largest number of identified peptides is typeset in boldface.

Sample ID	Max Quant	MS-GF+	Percolator	PLATO
2014-03-04-mel3p1	2533	2967	3748	**3898**
2014-03-04-mel3p2	2770	3341	**4467**	4412
2014-03-06-mel3p1	2441	2704	3632	**3681**
2014-03-06-mel3p2	2594	3140	**4271**	4157
2014-03-04-mel4p1	2634	3108	3929	**4073**
2014-03-04-mel4p2	1765	3073	**4004**	3757
2014-03-06-mel4p1	2401	2824	3526	**3682**
2014-03-06-mel4p2	1745	2694	3856	**3954**
2014-03-05-mel5p1	3010	2517	3742	**3923**
2014-03-05-mel5p2	3342	2561	4023	**4127**
2014-03-06-mel5p1	2934	2643	3929	**4059**
2014-03-06-mel5p2	3060	2592	**4070**	4014
2014-03-05-mel8p1	3375	3057	4006	**4297**
2014-03-05-mel8p2	3764	2976	4511	**4556**
2014-03-06-mel8p1	3331	3023	4375	**4454**
2014-03-06-mel8p2	4139	3444	4801	**4839**
2014-03-04-mel12p1	1948	1601	2724	**2870**
2014-03-04-mel12p2	2013	1408	2855	**3121**
2014-03-06-mel12p1	2004	1301	**3028**	3005
2014-03-06-mel12p2	2628	1601	2942	**3356**

combined PLATO with the open-source MS-GF+ database search engine [28], and used it to rescore peptide-spectrum matches (PSMs) using MS-GF+ features such as the match-score and spectrum charge, along with sequence defined features such as amino-acid composition and context. The use of PLATO increases the number of identified peptides at a fixed false discovery rate (FDR) compared to both model-based database search engines MS-GF+ and MaxQuant as well as the Percolator method, an existing rescoring approach based on support vector machines [29].

We have made available user-friendly web-based tools for peptide identification from MS/MS data by running the MS-GF+ and Percolator algorithms under the "Immunopeptidomics" section of the GeNeo Galaxy toolbox for Genomics Guided Neoepitope Prediction [8]. More information about these tools is provided in the appendix. A Python script that can be used to run PLATO on the output files generated by MS-GF+ is also available at github.com/esherafat/PLATO. Integration of PLATO into the GeNeo toolbox [8] is ongoing.

In future work we plan to assess PLATO's robustness to intra-tumor heterogeneity using large-scale exome sequencing datasets such as [30] and explore further improvements in peptide identification accuracy by incorporating additional features in the PLATO search. Finally, we plan to explore supervised and semi-supervised methods for predicting TRMNs. Improving TRMN prediction accuracy is critical for enabling routine clinical use of cancer vaccines since it is impractical to experimentally assess all candidate neoepitopes prior to vaccination [31].

4 Conclusion

In this paper we introduced PLATO, a novel semi-supervised approach to improving accuracy of model-based classifiers. PLATO generates a set of high confidence positive calls by applying a stringent filter to model-based predictions, then rescores remaining candidates by using positive-unlabeled learning. PLATO

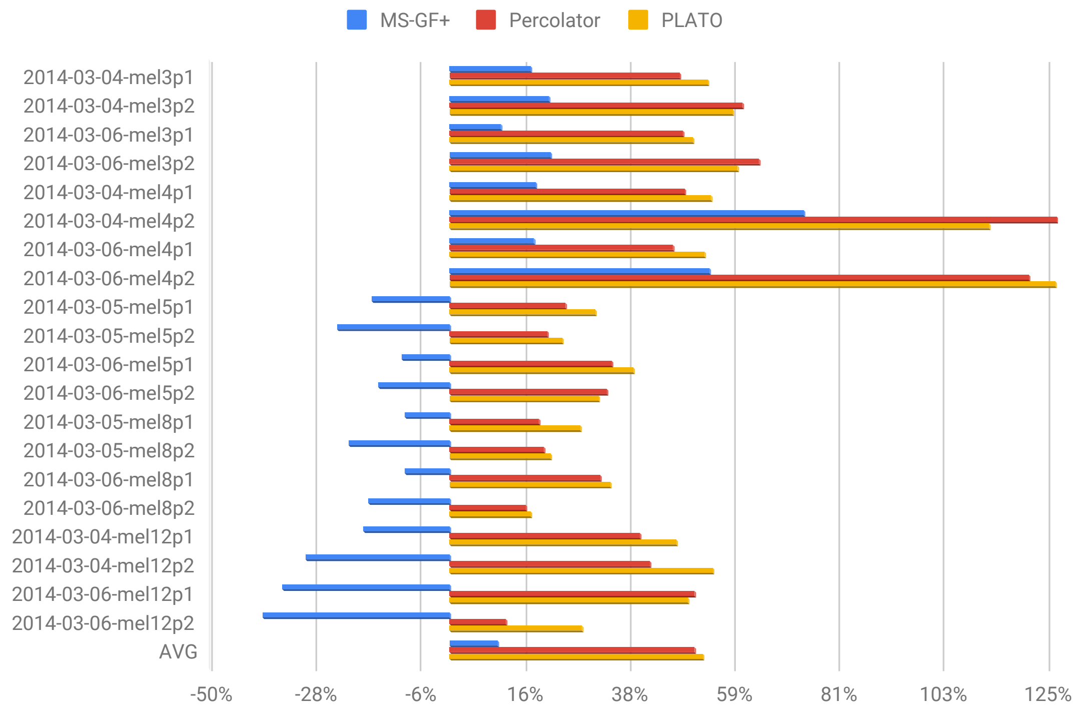


Figure 6: Percentage increase in the number of identified peptides over MaxQuant results reported in [27] using 1% FDR on the 20 MS/MS datasets from Table 2

further integrates AutoML hyper-parameter tuning, classification threshold selection based on spies, and bootstrapping to achieve robust performance on clinical samples with large patient-to-patient variation. Although the PU-learning framework implemented by PLATO is broadly applicable, in this paper we focused on its application and evaluation in the context of two problems arising in personalized cancer immunotherapy: somatic variant calling from matched tumor-normal exome sequencing data and peptide identification from immunopeptidomic MS/MS data. This allowed us to leverage the ability to conduct experimental validation of somatic variant calls as part of an ongoing clinical trial and rely on well-established techniques for controlling false discovery rate based on template-decoy competition in the case of peptide identification from MS/MS data. Experimental results on real datasets show improved PLATO performance compared to model-based approaches for both applications.

5 Methods

5.1 Positive-unlabeled learning

Semi-supervised learning is used when available training data is a combination of labeled and unlabeled samples. The key idea of semi-supervised learning is to use the unlabeled examples to modify, refine or prioritize the hypotheses derived from the labeled data alone. Positive-unlabeled learning is an important subcategory of semi-supervised learning, where only unlabeled and positive samples are available. One popular technique for PU learning is to predict a set of likely negatives among the unlabeled samples and then apply standard supervised machine learning methods to the set of positives and likely negatives. The PU learning framework implemented in PLATO is illustrated in Figure 2. Below we detail the key steps of this workflow.

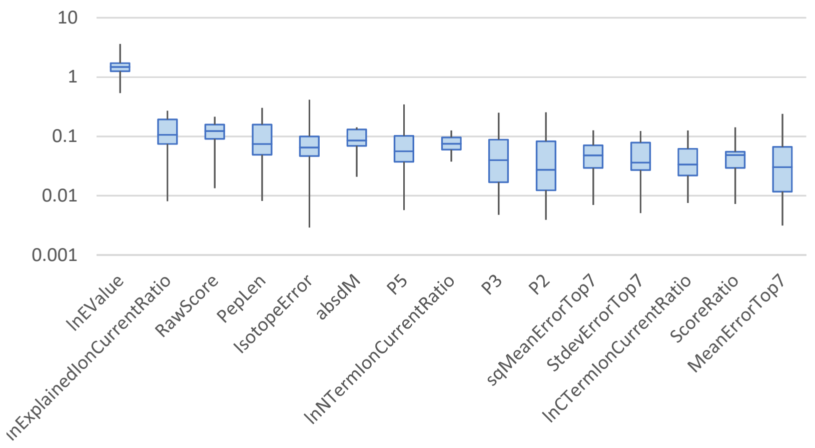


Figure 7: Boxplots of feature importance values (displayed on a logarithimic scale) for PLATO peptide identification experiments on the 20 MS/MS datasets from Table 2

5.1.1 Feature extraction and imputation

For variant calling, the sets P and U were generated from the output of the Consensus Caller Cross-Platform (CCCP) pipeline [23]. The set of positives was taken to be the set of variants passing the 2CP filter [26] that comes with the CCCP pipeline, and all other SNV candidates were included in U. Both positive and unlabeled samples were represented using a total of 110 features, 52 extracted from the output of the CCCP pipeline (see Table S3) and 58 generated using SomaticSeq [16] from the BAM files containing Illumina tumor and normal exome alignments (full list included in the appendix). This broad range of features included somatic variant calls made by the two somatic variant callers integrated in CCCP (SNVQ [24] and Strelka [25]), the coverage in tumor and normal samples, variant allele frequency, strand bias, membership in the list of common polymorphisms catalogued in the dbSNP database [32], average base and alignment quality, genomic region mappability, etc. The unfiltered output of CCCP includes a large percentage of missing values, typically due to low read coverage from one of the sequencing technologies. To deal with these missing values, prior to performing informed undersampling we removed the samples and features for which more than half of the corresponding entries were missing. Additionally, for the remaining samples we imputed missing features using the rfImpute function implemented by the randomForest CRAN package [33].

For peptide identification from MS/MS data, P and U were generated from the list of best *peptide-spectrum matches* (PSMs) generated using the MS-GF+ search engine for each spectrum (see the appendix for details). P was taken to be the set of PSMs identified by MS-GF+ at a False Discover Rate (FDR) cutoff of 1% (as estimated by target-decoy competition, see below), while U consisted of the remaining PSMs. PLATO was run using 27 features extracted from the MS-GF+ output (see Table S4). No imputation was performed for the MS/MS data.

5.1.2 Informed undersampling and feature selection

In both of our applications (SNV calling from matched tumor-normal sequencing data and peptide identification from tandem mass-spec data) the number of unlabeled samples vastly exceeds the number of labeled positives. For example, cancer datasets have a typical unlabeled:positive ratio of 1000:1 or higher. The vast majority of unlabeled examples are *a priori* expected to be true negatives (sequencing errors or germline variants). Ideally, we would like to train a classifier that predicts with high accuracy both the minority and the majority class. However, most classifiers tend to over-predict the majority class when they are trained with imbalanced data. One solution to this issue is to generate a balanced training dataset by using undersampling.

In PLATO we use undersampling to create a balanced training dataset consisting of the positive samples and an equally-sized set of likely negatives selected from the unlabeled samples. Due to the high imbalance in the unlabeled data, randomly sampling from the unlabeled samples is likely to produce a set consisting mostly of negative samples. However, some positive samples are also likely to be picked, and the randomly selected points may not be very well-separated from positive samples in the underlying feature space. Therefore our approach is to use *informed undersampling*, where we use the positive samples to inform the selection of likely negatives from the unlabeled set. Specifically, given sets P and U of positive and unlabeled samples, we generate the set $N \subseteq U$ of likely negatives as $N = \bigcup_{i=1}^{b} E_i$, where $|E_i| = |P|/b$. Each set E_i is computed by randomly selecting a batch of m samples from U, computing the average distance of each sample to the samples in P, and including in E_i the $|P|/b$ unlabeled samples with the greatest average distance. Since the data has both categorical and numerical features, the Gower distance is chosen as the distance measure. We chose to generate the likely negative set N by sampling multiple batches since the majority class might not be homogeneous (e.g., for SNV calling the negatives may represent sequencing errors or germline variants), and using multiple batches increases the chance of selecting representative samples from all regions of the majority class. For all experiments reported in this paper we used $b = 10$ and $m = |P|$. We did not conduct extensive empirical evaluation of these choices, but reasoned that they provide a good tradeoff between having enough subsamples to give a good representation of the search space and keeping the computational costs low by avoiding too many pairwise distance computations.

For somatic variant calling, once a balanced training dataset is generated by informed undersampling, PLATO uses a random forest classifier to rank all extracted features and selects the features with above median rank. This feature selection approach falls under the category of embedded methods, and is often used to enhance generalization and reduce running time of subsequent model training. We chose to use random forest-based feature selection over alternatives such as unsupervised dimensionality reduction methods like Principal Component Analysis (PCA) since the method works well with both numeric and categorical features and retains interpretability. Random forest-based feature selection is also highly scalable. This is an important consideration in PLATO, which performs this step for multiple bootstrap samples to increase classification robustness, as detailed below. Performing feature selection independently for each bootstrap also reduces the risk of overfitting, as different sets of features may be selected for different bootstrap runs.

Due to the lower number of available features, no feature selection was performed for the MS/MS data.

5.1.3 Bootstrapping and spy-based cutoffs

For robustness, PLATO implements PU-learning based on informed undersampling within a bootstrapping framework and implements a scheme of automatic classification threshold selection based on spies (see the flowchart in Figure 2). In each bootstrap iteration PLATO performs the following steps:

- Selects a set of likely negative equal to the size of 90% of positive data points using the informed undersampling method described above.
- Creates a training set by combining 90% of positive samples with the selected set of likely negatives.
- Builds a classifier using AutoML using this training set.
- Applies the classifier to the 10% of positive samples that were not included in training ("spy" samples) along with the other unlabeled samples.
- Classifies an unlabeled sample as positive, if its score is higher than the minimum score of the spy samples.

The above steps are repeated a user specified number of times (N bootstraps). A sample in U is finally classified as positive if it scores higher than the spy samples in a user-selected percentage of bootstrap runs, otherwise its final classification is negative. The idea of using "spies" was initially introduced in the text classification context [34]. As shown in the Results section, using automatically selected classification thresholds based on spies results in similar or better performance on clinical datasets than using the default classifier threshold, independent of the bootstrap support.

In general, model selection and hyperparameter tuning are complex tasks. Since exhaustively evaluating all combinations is unfeasible, we used the AutoML service integrated in Microsoft Azure to efficiently search the model space. In AutoML the user has the choice of executing an experiment on a local PC, a VM in the cloud, or a large cluster. In this work, we stored the data and executed all experiments locally. In each bootstrap of PLATO, AutoML was run on the balanced training dataset consisting of 90% of positives and an equally sized set of likely negatives generated by informed undersampling by selecting the experiment type as classification, defining the cross-validation scheme as 10-fold cross validation, and the primary metric as accuracy. For each experiment type, AutoML generates a set of initial pipeline parameters and executes a number of experiments with different parameters. In each experiment, it measures the primary metric using cross-validation and picks a new set of pipeline parameters until it reaches a threshold on execution time or the number of experiments. In the end, it builds an ensemble of different models to achieve optimal performance on the test set. Both voting and stack ensemble classifiers are currently supported. By default, they appear as the final iterations of each run. In order to have a powerful ensemble, AutoML initializes a list of up to five best scoring models (checking that their scores are within 5% of the best score) using the Caruana et al. algorithm [35]. In subsequent iterations, a new model is added to an existing ensemble only if it improves its accuracy based on the user selected metric. The voting ensemble classifier in AutoML uses soft-voting and makes predictions based on a weighted average of predicted class probabilities. The stack ensemble classifier has a two-layer implementation. It takes the same models as the voting ensemble as the first layer, and the second layer trains a meta-model to find the optimal combination of models from the first layer. The default meta-model for classification tasks in AutoML is LogisticRegression. In our experiments, the algorithms that were run through AutoML iterations included BernoulliNaiveBayes, ExtremeRandomTrees, LightGBM, RandomForest, and SGD. However, the best-fitted models selected by AutoML were always Voting and Stack ensemble classifiers. For example, in the peptide identification problem the Voting ensemble was chosen as the best model 90% of the time and the Stack ensemble was selected 10% of the time. We have chosen 10 as the number of AutoML iterations per bootstrap based on preliminary experiments showing that more than 10 iterations yield diminishing improvements in accuracy.

5.2 Processing mass-spectrometry data and assessing the false discovery rate

RAW files were converted to MGF format using RawConverter 1.1.0.19 [36]. We used the MS-GF+ search engine [28] to search the MGF files against the human proteome, with Unspecific Cleavage, and the TDA (Target-Decoy Analysis) option turned on. The decoy database generated by MS-GF+, consisting of reversed target peptides, was concatenated with the targets for FDR estimation. The MS-GF+ MZID output files were converted to PIN (Percolator INput) format using the `msgf2pin` utility from Percolator. The PIN files were post-processed using Percolator as well as PLATO. Further details on MS-GF+ and Percolator settings can be found in the appendix.

For estimating the false discovery rate (FDR) of peptide identifications from MS/MS data we adopted the commonly used *target-decoy competition* (TDC) approach. The first decision that needs to be made when using TDC is whether or not to search the target and decoy databases separately, or to concatenate them before searching. In the latter setting, each spectrum is matched with either a target or a decoy; in this way, the targets and decoys compete to match with each spectrum. As advocated in [37], we use concatenated searches in this study. Assuming that a higher score is better, for concatenated searches the FDR at a certain score threshold t can be estimated as in [29, 38]:

$$FDR(t) = \frac{1 + \text{Number of decoy PSMs with score } \geq t}{\text{Number of target PSMs with score } \geq t} \tag{1}$$

We control the FDR at level Q by finding the lowest score threshold t such that $FDR(t) < Q$, and only taking target PSMs with score greater than or equal to t. In this study, we controlled FDR at 1%.

Unfortunately, controlling the FDR at a level of α for PSMs does not imply control at the same level for peptides. As discussed in [39], a peptide present in the sample will, on average, be matched by a greater number of spectra than an absent peptide. To address this, if a peptide matches multiple spectra, we eliminate all but the best scoring PSM for that peptide. Once we have "uniquified" the peptides, we can apply the same q-value cutoff procedure as for PSMs. To ensure reproducibility and fair comparisons, we created a Galaxy tool to control FDR, publicly available at neo.engr.uconn.edu/?tool_id=FDR_custom_filter, and used it to filter the results of all compared methods (MS-GF+, Percolator, and PLATO) for which raw search results were generated as part of our empirical evaluation. More details on the precise procedure for controlling FDR at both the PSM and Peptide level as well as the Galaxy tool can be found in the appendix.

References

[1] Schumacher, T.N., Schreiber, R.D.: Neoantigens in cancer immunotherapy. Science **348**(6230), 69–74 (2015). doi:10.1126/science.aaa4971

[2] Srivastava, P.K.: Neoepitopes of cancers: Looking back, looking ahead. Cancer Immunology Research **3**(9), 969–977 (2015). doi:10.1158/2326-6066.CIR-15-0134

[3] Castle, J.C., Kreiter, S., Diekmann, J., Löwer, M., van de Roemer, N., de Graaf, J., Selmi, A., Diken, M., Boegel, S., Paret, C., Koslowski, M., Kuhn, A.N., Britten, C.M., Huber, C., Türeci, Ö., Sahin, U.: Exploiting the mutanome for tumor vaccination. Cancer Research **72**(5), 1081–1091 (2012). doi:10.1158/0008-5472.CAN-11-3722

[4] Duan, F., Duitama, J., Seesi, S.A., Ayres, C., Corcelli, S., Pawashe, A., Blanchard, T., McMahon, D., Sidney, J., Sette, A., Baker, B., Mandoiu, I.I., Srivastava, P.K.: Genomic and bioinformatic profiling of mutational neo-epitopes reveals new rules to predict anti-cancer immunogenicity. Journal of Experimental Medicine **211**(11), 2231–2248 (2014)

[5] Gubin, M.M., Zhang, X., Schuster, H., Caron, E., Ward, J.P., Noguchi, T., Ivanova, Y., Hundal, J., Arthur, C.D., Krebber, W.-J., Mulder, G.E., Toebes, M., Vesely, M.D., Lam, S.S.K., Korman, A.J., Allison, J.P., Freeman, G.J., Sharpe, A.H., Pearce, E.L., Schumacher, T.N., Aebersold, R., Rammensee, H.-G., Melief, C.J.M., Mardis, E.R., Gillanders, W.E., Artyomov, M.N., Schreiber, R.D.: Checkpoint blockade cancer immunotherapy targets tumour-specific mutant antigens. Nature **515**(7528), 577–81 (2014). doi:10.1038/nature13988

[6] Yadav, M., Jhunjhunwala, S., Phung, Q.T., Lupardus, P., Tanguay, J., Bumbaca, S., Franci, C., Cheung, T.K., Fritsche, J., Weinschenk, T., Modrusan, Z., Mellman, I., Lill, J.R., Delamarre, L.: Predicting immunogenic tumour mutations by combining mass spectrometry and exome sequencing. Nature **515**(7528), 572–576 (2014)

[7] Tumor Neoantigen Selection Alliance (TESLA) `https://www.parkerici.org/research-project/tumor-neoantigen-selection-alliance-tesla/`, accessed on 07/24/2020

[8] GeNeo: Bioinformatics Toolbox for *Ge*nomics Guided *Neo*epitope Prediction. `https://neo.engr.uconn.edu`, accessed on 07/24/2020

[9] Hundal, J., Carreno, B.M., Petti, A.A., Linette, G.P., Griffith, O.L., Mardis, E.R., Griffith, M.: pVAC-Seq: A genome-guided in silico approach to identifying tumor neoantigens. Genome Medicine **8**(1), 11 (2016). doi:10.1186/s13073-016-0264-5

[10] Zhou, Z., Lyu, X., Wu, J., Yang, X., Wu, S., Zhou, J., Gu, X., Su, S. Z. amd Chen: TSNAD: an integrated software for cancer somatic mutation and tumour-specific neoantigen detection. Royal Society Open Science **4** (2017)

[11] Bjerregaard, A.M., Nielsen, M., Hadrup, S.R., Szallasi, Z., Eklund, A.C.: MuPeXI: prediction of neoepitopes from tumor sequencing data. Cancer Immunology, Immunotherapy (2017)

[12] Ebrahimi-Nik, H., Michaux, J., Corwin, W.L., Keller, G.L.J., Shcheglova, T., Pak, H.S., Coukos, G., Baker, B.M., Mandoiu, I.I., Bassani-Sternberg, M., Srivastava, P.K.: Mass spectrometry driven exploration reveals nuances of neoepitope-driven tumor rejection. JCI Insight **4**(14) (2019)

[13] Krøigård, A., Thomassen, M., Lænkholm, A.-V., Kruse, T., Larsen, M.: Evaluation of nine somatic variant callers for detection of somatic mutations in exome and targeted deep sequencing data. P L o S One **11**(3) (2016). doi:10.1371/journal.pone.0151664

[14] Cai, L., Yuan, W., Zhang, Z., He, L., Chou, K.-C.: In-depth comparison of somatic point mutation callers based on different tumor next-generation sequencing depth data. Scientific Reports **6**, 36540 (2016). doi:10.1038/srep36540

[15] Spinella, J.-F., Mehanna, P., Vidal, R., Saillour, V., Cassart, P., Richer, C., Ouimet, M., Healy, J., Sinnett, D.: SNooPer: a machine learning-based method for somatic variant identification from low-pass next-generation sequencing. BMC genomics **17**(1), 912 (2016)

[16] Fang, L.T., Afshar, P.T., Chhibber, A., Mohiyuddin, M., Fan, Y., Mu, J.C., Gibeling, G., Barr, S., Asadi, N.B., Gerstein, M.B., *et al.*: An ensemble approach to accurately detect somatic mutations using SomaticSeq. Genome biology **16**(1), 197 (2015)

[17] Fan, Y., Xi, L., Hughes, D.S., Zhang, J., Zhang, J., Futreal, P.A., Wheeler, D.A., Wang, W.: MuSE: accounting for tumor heterogeneity using a sample-specific error model improves sensitivity and specificity in mutation calling from sequencing data. Genome biology **17**(1), 178 (2016)

[18] Huang, W., Guo, Y.A., Muthukumar, K., Baruah, P., Chang, M.M., Skanderup, A.J.: SMuRF: Portable and accurate ensemble prediction of somatic mutations. Bioinformatics (Oxford, England) (2019)

[19] Ramachandran, A., Li, H., Klee, E., Lumetta, S.S., Chen, D.: Deep learning for better variant calling for cancer diagnosis and treatment. In: Proceedings of the 23rd Asia and South Pacific Design Automation Conference, pp. 16–21 (2018). IEEE Press

[20] Sahraeian, S.M.E., Liu, R., Lau, B., Podesta, K., Mohiyuddin, M., Lam, H.Y.: Deep convolutional neural networks for accurate somatic mutation detection. Nature communications **10**(1), 1041 (2019)

[21] Poplin, R., Chang, P.-C., Alexander, D., Schwartz, S., Colthurst, T., Ku, A., Newburger, D., Dijamco, J., Nguyen, N., Afshar, P.T., *et al.*: A universal SNP and small-indel variant caller using deep neural networks. Nature biotechnology **36**(10), 983 (2018)

[22] Ainscough, B.J., Barnell, E.K., Ronning, P., Campbell, K.M., Wagner, A.H., Fehniger, T.A., Dunn, G.P., Uppaluri, R., Govindan, R., Rohan, T.E., *et al.*: A deep learning approach to automate refinement of somatic variant calling from cancer sequencing data. Nature genetics **50**(12), 1735 (2018)

[23] Consensus Caller Cross-Platform (CCCP). `https://neo.engr.uconn.edu/tool_runner?tool_id=CCCP`, accessed on 07/24/2020

[24] Duitama, J., Srivastava, P.K., Mandoiu, I.I.: Towards accurate detection and genotyping of expressed variants from whole transcriptome sequencing data. BMC Genomics **13(Suppl 2):S6** (2012)

[25] Saunders, C.T., Wong, W.S.W., Swamy, S., Becq, J., Murray, L.J., Cheetham, R.K.: Strelka: accurate somatic small-variant calling from sequenced tumor-normal sample pairs. Bioinformatics **28**(14), 1811 (2012)

[26] 2CP Filter. `https://neo.engr.uconn.edu/tool_runner?tool_id=CCCP_Filter`, accessed on 07/24/2020

[27] Bassani-Sternberg, M., Bräunlein, E., Klar, R., Engleitner, T., Sinitcyn, P., Audehm, S., Straub, M., Weber, J., Slotta-Huspenina, J., Specht, K., *et al.*: Direct identification of clinically relevant neoepitopes presented on native human melanoma tissue by mass spectrometry. Nature communications **7**, 13404 (2016)

[28] Kim, S., Pevzner, P.: MS-GF+ makes progress towards a universal database search tool for proteomics. Nature communications **5**, 5277 (2014). doi:10.1038/ncomms6277

[29] Käll, L., Canterbury, J.D., Weston, J., Noble, W.S., MacCoss, M.J.: Semi-supervised learning for peptide identification from shotgun proteomics datasets. Nature methods **4**(11), 923 (2007)

[30] Shi, W., Ng, C.K.Y., Lim, R.S., Jiang, T., Kumar, S., Li, X., Wali, V.B., Piscuoglio, S., Gerstein, M.B., Chagpar, A.B., Weigelt, B., Pusztai, L., Reis-Filho, J.S., Hatzis, C.: Reliability of whole-exome sequencing for assessing intratumor genetic heterogeneity. Cell Reports **25**(6), 1446–1457 (2018). doi:10.1016/j.celrep.2018.10.046

[31] Garcia-Garijo, A., Fajardo, C.A., Gros, A.: Determinants for neoantigen identification. Frontiers in Immunology **10**, 1392 (2019). doi:10.3389/fimmu.2019.01392

[32] Sherry, S.T., Ward, M.-H., Kholodov, M., Baker, J., Phan, L., Smigielski, E.M., Sirotkin, K.: dbSNP: the NCBI database of genetic variation. Nucleic Acids Research **29**(1), 308 (2001). doi:10.1093/nar/29.1.308

[33] Liaw, A., Wiener, M.: Classification and regression by randomForest. R News **2**(3), 18–22 (2002)

[34] Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: ICML, vol. 2, pp. 387–394 (2002). Citeseer

[35] Caruana, R., Niculescu-Mizil, A., Crew, G., Ksikes, A.: Ensemble selection from libraries of models. In: Proceedings of the Twenty-first International Conference on Machine Learning, p. 18 (2004)

[36] He, L., Diedrich, J., Chu, Y.-Y., Yates III, J.R.: Extracting accurate precursor information for tandem mass spectra by RawConverter. Analytical chemistry **87**(22), 11361–11367 (2015)

[37] Elias, J.E., Gygi, S.P.: Target-decoy search strategy for increased confidence in large-scale protein identifications by mass spectrometry. Nature methods **4**(3), 207 (2007)

[38] Kall, L.: Personal communication

[39] Granholm, V., Navarro, J.F., Noble, W.S., Käll, L.: Determining the calibration of confidence estimation procedures for unique peptides in shotgun proteomics. Journal of Proteomics **80**, 123–131 (2013). doi:10.1016/j.jprot.2012.12.007

[40] Granholm, V., Kim, S., Navarro, J.C., Sjolund, E., Smith, R.D., Kall, L.: Fast and accurate database searches with MS-GF+ percolator. Journal of proteome research **13**(2), 890–897 (2013)

S1 Supplementary Methods, Figures, and Tables

S1.1 MS-GF+ and percolator commands and settings

For the experiments reported in the paper we used MS-GF+ version `v2020.03.14`. Detailed command lines and parameters are provided below.

S1.1.1 MS-GF+ indexing command

When searching against a proteome multiple times the MS-GF+ developers recommend using the BuildSA tool to index the proteome. Note that unlike the `tide-index` command in Crux, MS-GF+ indexing does not actually generate the peptide database to be searched. Rather, it builds suffix arrays for the proteins; this is why we do not specify any enzyme settings at this stage. The MS-GF+ index command we used was:

```
java -Xmx3500M -cp MSGFPlus.jar edu.ucsd.msjava.msdbsearch.BuildSA -d FASTA_INPUT -tda 2
```

S1.1.2 MS-GF+ search settings

When running the MS-GF+ search, our pipeline used commands of the following form:

```
    java -Xmx10000M -jar MSGFPlus.jar -ignoreMetCleavage 1 -s MGF_INPUT
    -d FASTA_INPUT -tda 1 -o MZID_OUTPUT -addFeatures 1 -m FRAGMENTATION -e 0 -inst INSTRUMENT
-mod MOD_FILE
```

A summary of the MS-GF+ options is provided in Table S1.

S1.1.3 msgf2pin settings

When running the msgf2pin utility to convert the MS-GF+ MZID output to a Percolator Input (PIN) file, our pipeline used the following command:

```
    msgf2pin MZID_INPUT -o PIN_OUTPUT -F FASTA_INPUT -e no_enzyme -P XXX_ -m 1 -z
```

Table S1: MS-GF+ search options.

Option	Value	Description
-Xmx	10000M	Tells the JVM to set the maximum heap size to 10GB
-ignoreMetCleavage	1	A (hidden) setting to turn off methionine cleavage [1]
-s	MGF_INPUT	The MGF File to search
-d	FASTA_INPUT	The FASTA file to search
-tda	1	Create (and search) a combined target-decoy database
-o	MZID_OUTPUT	Filename of MZID output
-addFeatures	1	Include additional features that Percolator will use; see [40] for more information
-m	3	Sets the fragmentation method; in this case it's 3, since the data was generated with HCD fragmentation
-e	0	Use non-specific cleavage when creating peptides to search
-inst	3	Sets the instrument type in this case, it's 3, since the data was generated with a Q-Exactive instrument
-mod	MOD_FILE	A file containing the post-translational modifications to include in the search. We used Cysteine Carbamidomethylation as a fixed modification since iodoacetamide was used in the experiments of [27].

Table S2: msgf2pin search options.

Option	Value	Description
	MZID_INPUT	Mzid file to convert to PIN
-o	PIN_OUTPUT	filename of the PIN output
-F	FASTA_INPUT	Proteome that was searched against
-e	no_enzyme	Type of enzyme used for *in silico* digestion. Used no_enzyme if there non-specific digestion.
-P	XXX_	In the Mzid file, each peptide is associated with at least one protein sequence. Each protein sequence has an accession, and decoy proteins have an accession that starts with "XXX_"
-m	1	Maximum number of matches per spectra
-z		Displays PTM as mass delta, rather than UNIMOD Accession

Table S2 gives a summary of the options used by this command. Note that we used a slightly modified version of msgf2pin, the source code of which can be found here: `https://github.com/mrForce/msgf2pin-PTM-Mass-Delta`. We added the "-z" option, so that post-translational modifications would be annotated with mass deltas, rather than UNIMOD accession codes. This is because, at one point, we were working with a version of Percolator that wasn't compatible with UNIMOD accession. Note that, although msgf2pin is part of the Percolator package, we used it as a standalone utility.

S1.1.4 Percolator command

Percolator commands were of the form:

```
crux percolator --output-dir OUTPUT_DIR PIN_INPUT
```

We are currently using Percolator version `3.02.0` in Crux version `3.20-d57cff`.

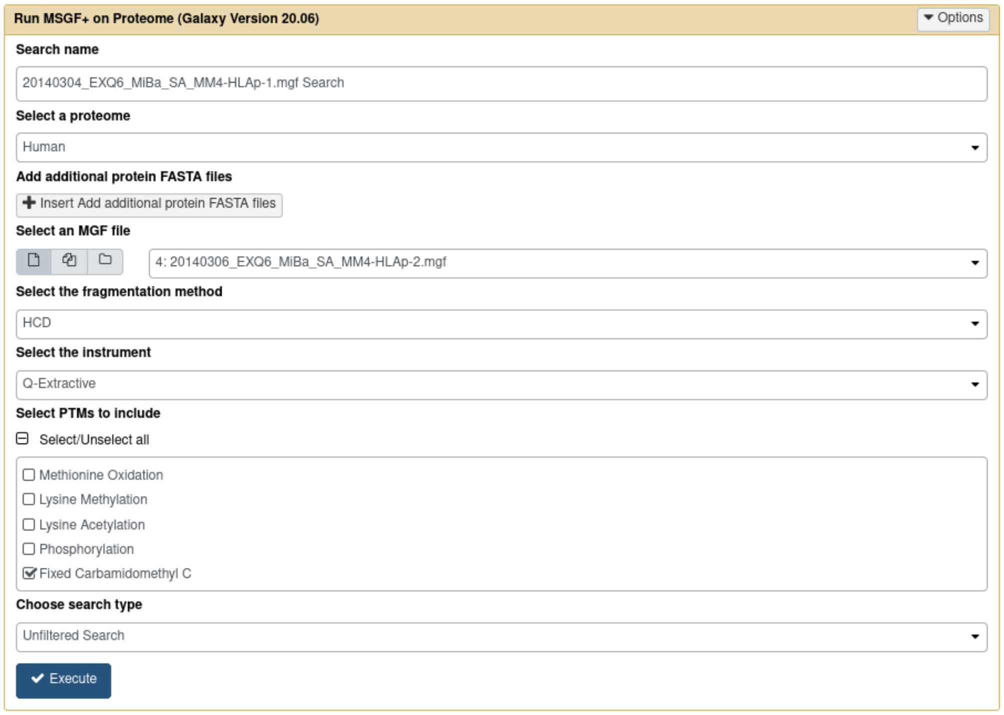
S1.2 Galaxy Search tool

We created a publicly available Galaxy tool that allows users to run MS-GF+ and Percolator through a web-based user-friendly interface. The tool can be accessed at `https://neo.engr.uconn.edu/?tool_id=msgfplus_runner`; the tool version used in this study was 20.06. Figure S1 shows a screenshot of the interface.

The Galaxy search tool supports two search types. The first is called "Unfiltered Search", where the selected MGF file is searched against the selected proteome (concatenated with any user provided FASTA files). The second type is called "Filtered Search". Briefly the base proteome (and any uploaded FASTA files) are broken into peptides with lengths between 8 and 13 amino acids. The user specifies a set of MHC-I/HLA-I alleles, and the peptides are scored using NetMHC. For each allele-length combination, the top k percent scoring peptides are used in the search, where k is a user specified parameter. In this study, we only used the Unfiltered Search.

The user can give the search a name, and select which proteome to search. For this study, we used a Uniprot Human proteome consisting of one protein per gene, which was downloaded from `ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Eukaryota/UP000005640_9606.fasta.gz` in April 2019. Currently, this is the only proteome offered for searching, though more proteomes will be added in the future.

[1]This setting is not mentioned in the MS-GF+ documentation. See this bug report for more information: `https://github.com/MSGFPlus/msgfplus/issues/51`


Run MSGF+ on Proteome (Galaxy Version 20.06)

Search name: 20140304_EXQ6_MiBa_SA_MM4-HLAp-1.mgf Search

Select a proteome: Human

Add additional protein FASTA files

Select an MGF file: 4: 20140306_EXQ6_MiBa_SA_MM4-HLAp-2.mgf

Select the fragmentation method: HCD

Select the instrument: Q-Extractive

Select PTMs to include

- [ ] Methionine Oxidation
- [ ] Lysine Methylation
- [ ] Lysine Acetylation
- [ ] Phosphorylation
- [x] Fixed Carbamidomethyl C

Choose search type: Unfiltered Search

Figure S1: Interface for the Galaxy MS/MS search tool.

The tool has four outputs. The first is the PIN file generated by msgf2pin, except with the second row removed. This second row contains information that Percolator needs, but is otherwise not useful to us. The second output are the target and decoy PSMs scored by Percolator. The third is a log file, which is useful for debugging (and also shows exactly how MS-GF+, msgf2pin and Percolator were ran). The fourth is an archive file, which contains, among other things, the MZID output from MS-GF+, the PIN file passed to Percolator, and the output files of the Percolator run. It also contains the FASTA that MS-GF+ searched.

Published Galaxy histories including runs for the 20 MS/MS melanoma datasets analyzed in this paper (grouped by patient) are available at:

- `https://neo.engr.uconn.edu/u/jordan/h/bassani-mel3-public`
- `https://neo.engr.uconn.edu/u/jordan/h/bassani-mel4-public`
- `https://neo.engr.uconn.edu/u/jordan/h/bassani-mel5-public`
- `https://neo.engr.uconn.edu/u/jordan/h/bassani-mel8-public`
- `https://neo.engr.uconn.edu/u/jordan/h/bassani-mel12-public`

S1.3 FDR Filtering

To fairly assess the number of discoveries each tool makes at a given FDR cutoff, we wrote a Galaxy tool to control FDR at both the PSM and Peptide level. The tool takes as input a tab-seperated value file, and the user specifies which columns contain the peptide, score and label (target or decoy), and the score direction (whether a bigger score is better or worse), as well as an FDR cutoff. The tool will have one output for PSM level FDR filtering, and another for Peptide level FDR filtering. For the Peptide level, it uniquifies the peptides by selecting the best scoring PSM for each peptide, and discards poorer scoring PSMs for that peptide. From then on, the procedure is the same for PSM or Peptide level FDR filtering:

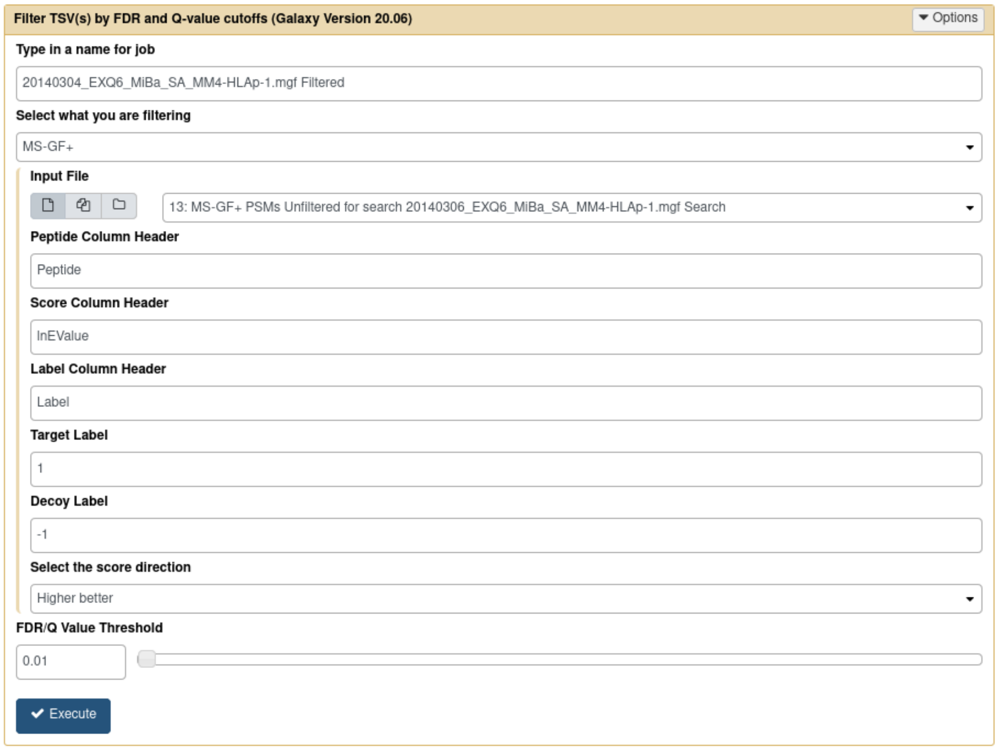
```
 1: $groups \leftarrow$ Group PSMs by score
 2: $sortedGroups \leftarrow$ Sort groups by score, from best to worst
 3: $numDecoys \leftarrow 0$
 4: $numTargets \leftarrow 0$
 5: $endIndex \leftarrow -1$
 6: $i \leftarrow 0$
 7: for $score, group \leftarrow$ groups do
 8:     for $psm \leftarrow$ group do
 9:         if $psm$ is target then $numTargets \leftarrow numTargets + 1$
10:         else if $psm$ is decoy then $numDecoys \leftarrow numDecoys + 1$
11:         end if
12:     end for
13:     if $numTargets = 0$ then
          $fdr \leftarrow 1$
14:     else
          $fdr \leftarrow \frac{numDecoys+1}{numTargets}$
15:     end if
16:     if $fdr < \alpha$ then
17:         $endIndex \leftarrow i$
18:     end if
19:     $i \leftarrow i + 1$
20: end for
```

The target PSMs in the groups up to *endIndex* are then controlled at FDR-level α. The grouping is necessary because frequently, there will be PSMs with the same score, and they must either be accepted or rejected together as a group. For Percolator, we used the "percolator score" column as the score. For MS-GF+, we used the $lnEValue$ in the msgf2pin output. This is simply the negative logarithm of a PSM's E-Value. Note that MS-GF+ provides Q-values, which can also be used for FDR control; however, their Q-values are computed based on the Spectral E-Value. The reason for this discrepancy is that we forked Percolator version 3.04 to create the custom version of msgf2pin (see the "msgf2pin settings" subsection above), and that version wasn't able to output both $lnEValue$ and $lnSpecEValue$.

As for MS/MS searches, we created a Galaxy tool that allows users to run the FDR filter through a web-based user-friendly interface. The FDR filter tool (version 20.06) can be accessed at `https://neo.engr.uconn.edu/tool_runner?tool_id=FDR_custom_filter`; Figure S2 displays a screenshot of its user interface.

S1.4 PLATO feature descriptions

For SNV calling PLATO used 52 features generated by the CCCP pipeline (described in Table S3) along with the following 58 additional features extracted using SomaticSeq [16] from the BAM files containing Illumina tumor and normal exome alignments: Consistent_Mates, Inconsistent_Mates, MaxHomopolymer_Length, N_ALT_FOR, N_ALT_REV, N_DP, N_REF_FOR, N_REF_REV, nBAM_ALT_BQ, nBAM_ALT_Clipped_Reads, nBAM_ALT_Concordant, nBAM_ALT_Discordant, nBAM_ALT_MQ, nBAM_ALT_NM, nBAM_Clipping_FET, nBAM_Concordance_FET, nBAM_MQ0, nBAM_NM_Diff, nBAM_Other_Reads, nBAM_Poor_Reads, nBAM_REF_BQ, nBAM_REF_Clipped_Reads, nBAM_REF_Concordant, nBAM_REF_Discordant, nBAM_REF_MQ, nBAM_REF_NM, nBAM_StrandBias_FET, nBAM_Z_Ranksums_BQ, nBAM_Z_Ranksums_EndPos, nBAM_Z_Ranksums_MQ, SiteHomopolymer_Length, T_ALT_FOR, T_ALT_REV, T_DP, T_REF_FOR, T_REF_REV, tBAM_ALT_BQ, tBAM_ALT_Clipped_Reads, tBAM_ALT_Concordant, tBAM_ALT_Discordant, tBAM_ALT_MQ, tBAM_ALT_NM, tBAM_Clipping_FET, tBAM_Concordance_FET, tBAM_MQ0, tBAM_NM_Diff, tBAM_Other_Reads, tBAM_Poor_Reads, tBAM_REF_BQ, tBAM_REF_Clipped_Reads, tBAM_REF_Concordant, tBAM_REF_Discordant, tBAM_REF_MQ, tBAM_REF_NM, tBAM_StrandBias_FET, tBAM_Z_Ranksums_BQ, tBAM_Z_Ranksums_EndPos, tBAM_Z_Ranksums_MQ.



Filter TSV(s) by FDR and Q-value cutoffs (Galaxy Version 20.06)

Options

Type in a name for job: 20140304_EXQ6_MiBa_SA_MM4-HLAp-1.mgf Filtered

Select what you are filtering: MS-GF+

Input File: 13: MS-GF+ PSMs Unfiltered for search 20140306_EXQ6_MiBa_SA_MM4-HLAp-1.mgf Search

Peptide Column Header: Peptide

Score Column Header: lnEValue

Label Column Header: Label

Target Label: 1

Decoy Label: -1

Select the score direction: Higher better

FDR/Q Value Threshold: 0.01

Execute

Figure S2: Interface for the Galaxy FDR filter tool.

Table S3: CCCP output features used by PLATO for SNV calling.

Feature	Description
Platform	Platform supporting the call (Illumina, Proton, or Both)
Ref_allele	Reference allele
dbSNP	Yes/No common polymorphism according to dbSNP
Alt_in_dbSNP	Alternative allele in dbSNP
Alt_SNVQ_N_ILL	Alternative allele for the SNVQ call from normal exome Illumina alignments
Alt_SNVQ_T_ILL	Alternative allele for the SNVQ call from tumor exome Illumina alignments
Geno_SNVQ_T_ILL	Genotype for the SNVQ call from tumor exome Illumina alignments
Alt_Strelka_T_ILL	Alternative allele for the Strelka call based on tumor/normal exome Illumina alignments
Geno_Strelka_T_ILL	Genotype for the Strelka call based on tumor/normal exome Illumina alignments
SNV_in_SNVQ_N_ILL	Yes/No SNV called by SNVQ run on normal exome Illumina alignments
SNV_in_SNVQ_T_ILL	Yes/No SNV called by SNVQ run on tumor exome Illumina alignments
Som_SNVQ_T/N_ILL	Yes/No somatic SNV called by the SNVQ subtraction method based on tumor/normal exome Illumina alignments
Som_Strelka_T/N_ILL	Yes/No somatic SNV called by Strelka based on tumor/normal exome Illumina alignments
Total_Cov_N_ILL	Total coverage in normal exome Illumina alignments
A_Cov_N_ILL	Coverage of allele A in normal exome Illumina alignments
C_Cov_N_ILL	Coverage of allele C in normal exome Illumina alignments
G_Cov_N_ILL	Coverage of allele G in normal exome Illumina alignments

Continued on next page

Table S3 – *Continued from previous page*

Feature	Description
T_Cov_N_ILL	Coverage of allele T in normal exome Illumina alignments
Total_Cov_T_ILL	Total coverage in tumor exome Illumina alignments
A_Cov_T_ILL	Coverage of allele A in tumor exome Illumina alignments
C_Cov_T_ILL	Coverage of allele C in tumor exome Illumina alignments
G_Cov_T_ILL	Coverage of allele G in tumor exome Illumina alignments
T_Cov_T_ILL	Coverage of allele T in tumor exome Illumina alignments
Alt_SNVQ_N_ILL	Alternative allele for the SNVQ call from normal exome Proton alignments
Alt_SNVQ_T_ION	Alternative allele for the SNVQ call from tumor exome Proton alignments
Geno_SNVQ_T_ION	Genotype for the SNVQ call from tumor exome Proton alignments
Alt_Strelka_T_ION	Alternative allele for the Strelka call based on tumor/normal exome Proton alignments
Geno_Strelka_T_ION	Genotype for the Strelka call based on tumor/normal exome Proton alignments
SNV_in_SNVQ_N_ION	Yes/No SNV called by SNVQ run on normal exome Proton alignments
SNV_in_SNVQ_T_ION	Yes/No SNV called by SNVQ run on tumor exome Proton alignments
Som_SNVQ_T/N_ION	Yes/No somatic SNV called by the SNVQ subtraction method based on tumor/normal exome Proton alignments
Som_Strelka_T/N_ION	Yes/No somatic SNV called by Strelka based on tumor/normal exome Proton alignments
Total_Cov_N_ION	Total coverage in normal exome Proton alignments
A_Cov_N_ION	Coverage of allele A in normal exome Proton alignments
C_Cov_N_ION	Coverage of allele C in normal exome Proton alignments
G_Cov_N_ION	Coverage of allele G in normal exome Proton alignments
T_Cov_N_ION	Coverage of allele T in normal exome Proton alignments
Total_Cov_T_ION	Total coverage in tumor exome Proton alignments
A_Cov_T_ION	Coverage of allele A in tumor exome Proton alignments
C_Cov_T_ION	Coverage of allele C in tumor exome Proton alignments
G_Cov_T_ION	Coverage of allele G in tumor exome Proton alignments
T_Cov_T_ION	Coverage of allele T in tumor exome Proton alignments
Total_Cov_T_ILL_RNA	Total coverage in tumor RNA-Seq Illumina alignments
A_Cov_T_ILL_RNA	Coverage of allele A in tumor RNA-Seq Illumina alignments
C_Cov_T_ILL_RNA	Coverage of allele C in tumor RNA-Seq Illumina alignments
G_Cov_T_ILL_RNA	Coverage of allele G in tumor RNA-Seq Illumina alignments
T_Cov_T_ILL_RNA	Coverage of allele T in tumor RNA-Seq Illumina alignments
Total_Cov_T_ION_RNA	Total coverage in tumor RNA-Seq Proton alignments
A_Cov_T_ION_RNA	Coverage of allele A in tumor RNA-Seq Proton alignments
C_Cov_T_ION_RNA	Coverage of allele C in tumor RNA-Seq Proton alignments
G_Cov_T_ION_RNA	Coverage of allele G in tumor RNA-Seq Proton alignments
T_Cov_T_ION_RNA	Coverage of allele T in tumor RNA-Seq Proton alignments

For peptide identification PLATO used the 25 features listed in Table S4, which were extracted from the MS-GF+ output.

Table S4: Features used by PLATO for peptide identification from MS/MS data.

Feature	Description
`CalcMass`	Theoretical mass of peptide (sum of amino acid masses)
`Mass`	Spectrum precursor mass
`dM`	Theoretical mass minus observed mass
`absdM`	Absolute value of `dM`
`IsotopeError`	The number of additional neutrons in the peptide compared to the monoisotopic mass
`MeanErrorTop7`	Mean mass error of 7 most intense peaks
`sqMeanErrorTop7`	Square root of `MeanErrorTop7`
`StdevErrorTop7`	Standard deviation of mass errors of 7 most intense peaks
`Charge1, Charge2, Charge3`	Spectrum charge
`DeNovoScore`	Score of best scoring peptide for the spectrum. This is among all possible peptides, not just those in the database
`RawScore`	The PSM score assigned by MS-GF+
`Energy`	Difference between RawScore and DeNovoScore
`ScoreRatio`	Ratio of `RawScore` to maximum possible score (aka `DeNovoScore`)
`lnEValue`	Negative one times the natural logarithm of the database level E-value [40]. See Kim and Pevzner [28] for a detailed description of how E-value is calculated by MS-GF+
`lnExplainedIonCurrentRatio`	Logarithm of the total intensity of identified fragment ions divided by total intensity of all ions
`lnNTermIonCurrentRatio`	Logarithm of total intensity of identified N-terminal fragment ions divided by total intensity of all ions
`lnCTermIonCurrentRatio`	Logarithm of total intensity of identified C-terminal fragment ions divided by total intensity of all ions
`lnMS2IonCurrent`	Logarithm of sum of intensities of all fragment ions
`PepLen`	Peptide length
`P1` and `P6`	The amino acids before and after the peptide in its protein
`P2` and `P3`	The first two amino acids of the peptide
`P4` and `P5`	The last two amino acids of the peptide