## Title

Orchid: a novel management, annotation and machine learning framework for analyzing cancer mutations



## Journal

Bioinformatics, 34(6)

## ISSN

1367-4803

## Authors

Cario, Clinton L
Witte, John S

## Publication Date

2018-03-15

## DOI

10.1093/bioinformatics/btx709

Peer reviewed

*Bioinformatics*, 34(6), 2018, 936–942
doi: 10.1093/bioinformatics/btx709
Advance Access Publication Date: 2 November 2017
Original Paper

OXFORD

Genome analysis

# Orchid: a novel management, annotation and machine learning framework for analyzing cancer mutations

**Clinton L. Cario and John S. Witte***

Department of Epidemiology and Biostatistics, University of California, San Francisco, CA 94158, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on July 18, 2017; revised on October 27, 2017; editorial decision on October 30, 2017; accepted on October 31, 2017

## Abstract

**Motivation:** As whole-genome tumor sequence and biological annotation datasets grow in size, number and content, there is an increasing basic science and clinical need for efficient and accurate data management and analysis software. With the emergence of increasingly sophisticated data stores, execution environments and machine learning algorithms, there is also a need for the integration of functionality across frameworks.

**Results:** We present orchid, a python based software package for the management, annotation and machine learning of cancer mutations. Building on technologies of parallel workflow execution, in-memory database storage and machine learning analytics, orchid efficiently handles millions of mutations and hundreds of features in an easy-to-use manner. We describe the implementation of orchid and demonstrate its ability to distinguish tissue of origin in 12 tumor types based on 339 features using a random forest classifier.

**Availability and implementation:** Orchid and our annotated tumor mutation database are freely available at https://github.com/wittelab/orchid. Software is implemented in python 2.7, and makes use of MySQL or MemSQL databases. Groovy 2.4.5 is optionally required for parallel workflow execution.

**Contact:** JWitte@ucsf.edu

**Supplementary information:** Supplementary data are available at *Bioinformatics* online.

## 1 Introduction

Cancer is a complicated disease driven largely by genomic alterations. To better understand and characterize the genetic architecture underlying carcinogenesis, thousands of tumor genomes have been sequenced. This work has detected a large number of somatic mutations, gleaning meaningful biological insight such as identification of functional driver mutations in dozens of genes like KRAS, APC, P53, PI3K, SMAD4 (Vogelstein *et al.*, 2013) that are involved in many cancers.

A key challenge in the analysis of tumor genomes is how to interpret mutations with uncertain function. This is further complicated by the fact that many mutations may have no relevant function, but arise simply as artifacts of an unstable and mutated tumor genome (i.e. as passengers). To address these issues, several statistical and computational algorithms have been developed that attempt to prioritize or annotate mutations by finding genes with higher than expected mutation rates (Dees *et al.*, 2012; Lawrence *et al.*, 2013), by analyzing predicted functional effects of mutations (Choi *et al.*, 2012; Kumar *et al.*, 2009) and by exploiting interaction networks of protein pathways to infer relevant disrupting mutations (Subramanian *et al.*, 2005; Vandin *et al.*, 2012).

Recently a new class of methods inspired by machine learning paradigms have emerged that determine 'deleteriousness' of mutations in both general and cancer-specific contexts. These consist of models trained in evolutionary conservation (Kircher *et al.*, 2014; Quang *et al.*, 2015), protein sequence, domain and/or structural

information (Adzhubei *et al.*, 2010; Carter *et al.*, 2009), and parsimonious analysis of a broad range of tumor datasets (Kumar *et al.*, 2016).

Despite successful applications, there remain limitations to such statistical and machine learning approaches. For one, few methods annotate or score mutations that fall outside of coding regions despite the known regulatory importance of many intergenic bodies (e.g. enhancers, promoters, transcription factor binding sites, or microRNAs) (Raphael *et al.*, 2014). There are also issues with the collection, parsing and integration of tumor and annotation information that is scattered across dozens of databases and in a variety of formats, many of which are not suited for high-throughput analysis. Finally, methods that can score variants at a base-level resolution tend to be general in what they predict (e.g. evolutionary conservation), making more refined predictions difficult (e.g. likelihood of being a prostate cancer driver mutation).

To address some of these issues, we developed *orchid*, an open source tumor mutation management and machine learning analysis framework. Orchid makes the management, annotation and analysis of tumor mutations more programmatically elegant and computationally efficient by integrating mutation data with popular databases and python-based numeric and machine learning frameworks. Orchid is capable of accepting a wide assortment of feature types and is agnostic to the desired classification task, making it easy to build a variety of models quickly. Furthermore, it accepts and annotates mutations from any region of the genome, allowing for the analysis of non-coding mutations.

To demonstrate orchid, we applied it to the task of inferring cancer tissue-of-origin based upon copy number information and simple somatic mutations found in the genomes of 12 tumor types. This application highlights the value of orchid in generating models that can potentially be used in the diagnosis of metastatic tumors from which primary tumor cannot be located (called 'cancers of unknown primary' or CUPS), which represent 2–4% of all cancers (Pavlidis and Pentheroudakis, 2012), or in identifying tissue-of-origin from mutations found within cell-free DNA (cfDNA).

## 2 Materials and methods

We created an open-source mutation management and modeling software package called orchid which consists of the *orchid-db* script for loading and annotating mutations into a MySQL-like database system, and *orchid-ml*, a python module that interfaces with the popular python numeric analysis library, pandas (http://pandas.pydata.org/) and with scikit-learn (Pedregosa *et al.*, 2012) (http://scikit-learn.org/), a python framework for machine learning. Orchid has the ability to parse raw data in various common formats and can be used to generate annotated tumor mutational databases and models in as little as ten lines of code. A diagram of the orchid workflow is shown in Figure 1.

### 2.1 Orchid-db

#### *2.1.1* Datasets for tumor database generation

To build a tumor mutational database for subsequent supervised machine learning tasks, we first downloaded and collected raw tumor variants calls, copy number information and metadata for multiple tumor types as well as variant annotation data from several biological databases. For tumor data, we choose to make use of the International Cancer Genome Consortium (ICGC) given its extensive collection of tumor data across dozens of studies and tissue types. For annotation data, we hand selected biological features to represent a broad range of functional genomic annotations, prioritizing genome-wide annotation datasets when available. Data was populated into a MySQL or MemSQL database (see Supplementary Material) using the orchid software running on either a 2013 MacPro (OSX Sierra) or a PBS Cluster (Red Hat Linux v6.6). A MemSQL version of this database is available for public use; please see https://github.com/wittelab/orchid.

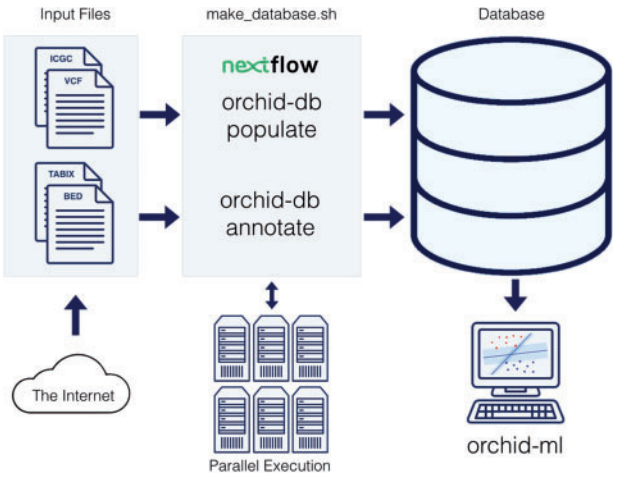


**Fig. 1.** Diagram of Orchid Workflow. The *make_database* shell script builds a database of annotated cancer mutations from raw source data using the *orchid-db populate* and *annotate* subscripts and can be run on a single computer or in a cluster environment. Afterwards, data can be quickly imported and analyzed with machine learning algorithms using *orchid-ml*

#### *2.1.2* Real ICGC mutations

From the ICGC data portal, we selected patients from release 25 with genome wide simple somatic mutation and/or copy number tumor data that were publically available (non PCAWG), for a total of 3604 individuals. We then excluded outliers by removing those individuals whose tumors had less than 10 or more than 30 000 mutations. Finally, we excluded tissues with fewer than 80 tumors and randomly sampled 80 from the remaining, resulting in 960 tumors from twelve cancer types (Bladder, Blood, Bone, Brain, Breast, Esophagus, Head and neck, Pancreas, Prostate, Skin, Stomach and Uterus). In total, 3 489 978 mutations were populated into the database with tissue means ranging from 281 for Bladder to 15 202 for Esophagus (Supplementary Fig. S1). Conceptually, we grouped this data into two levels of specificity for analysis: 1) the mutational level—where real mutations found within patients of a single tumor type are compared to mutations that might occur by chance through careful simulation; and 2) the tumoral level—where real mutations from patients (either of the same tumor type of different tumors) are compared to each other. Possible classification outcomes (i.e. labels) are 'observed' (or 'real') and 'simulated' for mutational level classifications, and any patient-level stratifier (e.g. tumor tissue-of-origin, stage, aggressiveness) for the tumoral level. The tissue-of-origin application presented in this paper represents classification of tumoral-level data.

#### *2.1.3* Biological features

We collected, downloaded and curated mutational annotation from 15 biological databases and annotation tools (Supplementary Table S1; http://wittelab.ucsf.edu/orchid). These features include functional annotation (*SnpEff*, Cingolani *et al.*, 2012); cancer gene network presence (*KEGG*, Kanehisa and Goto, 2000); phylogenetic

conservation (*phylop*, Pollard *et al.*, 2010); location within snoRNA and microRNA regions (*wgrna*, Griffiths-Jones, 2004); locations within predicted enhancers, promoters and transcription start sites (*segmentation*, Ernst and Kellis, 2012; Hoffman *et al.*, 2012; *rfecs*, Rajagopal *et al.*, 2013; *dbsuper*, Khan and Zhang, 2016; *encode*, Kundaje *et al.*, 2015); locations within DNAse I hypersensitivity sites (*dnase*, Thurman *et al.*, 2012); trinucleotide contexts (Alexandrov *et al.*, 2013); assorted composite scores (*funseq2*, Fu *et al.*, 2014; *cadd*, Kircher *et al.*, 2014; *dann*, Quang *et al.*, 2015); and various other measures (*targetscans*, Agarwal *et al.*, 2015; *remap*, Griffon *et al.*, 2015; *gwas*, MacArthur *et al.*, 2017).

There are a wide variety of suitable biological annotations and file types that can serve as mutation features. We therefore designed our annotation software tool to be flexible in the file formats it accepts. Features in tabix, bed, or wig file formats can be added to the modeling process with no modification while other formats can be integrated with minimal effort—namely by converting to bed format or by providing orchid with UNIX awk commands to pre- and post-parse feature lookup data.

All annotation and mutational data is based on genome coordinates from human reference sequence version GRCh37 (hg19). Data should be in the same coordinate system for database population. For convenience, GRCh38 (hg38) coordinates are also provided in our publically accessible database.

## 2.2 Orchid-ml

Orchid-ml exists as a standalone python module that can be imported into any python script. To load, transform, model and visualize mutation data, we designed the *MutationMatrix* object, an extension of the pandas *DataFrame* object. Transformations of the data include loading, encoding, imputing, feature scaling and feature selection. Modeling consists of selecting a prediction label and running orchid's built-in support vector machine (SVM) or random forest (RF) wrapper functions or any of the scikit-learn classifiers. Finally, visualization produces ROC curves, confusion matrices and other performance metric tables.

### *2.2.1* Loading and encoding

We implemented several functions to load and encode data as a *MutationMatrix*. The first, *load_mutations()*, will take a MySQL connection string for a database populated by orchid-db and load all (or a desired subset of) mutations and their basic associated metadata (chromosome, position, donor_id, sequence, etc.). The second, *load_features()*, will import all (or a desired subset of) annotation features. Finally, *encode()* will transform categorical features into numeric values so they can be properly modeled. This is accomplished through the specification of encoding strategies given as a dictionary to the function (strategies = {feature: strategy}). Choices for strategy are 'one-hot', 'binary', 'label' or 'rarity' (i.e. a feature value's frequency). Alternatively, or if not specified, orchid will use a one-hot encoder (see Supplementary Material for more details).

### *2.2.2* Collapsing

In some situations, it may be desirable to aggregate mutational level data to the tumoral level to compare tumor mutational profiles with each other. For this purpose, we created the *collapse()* function to aggregate feature values within each patient (i.e. tumor) using feature median or mean values. In practice this is can be done with any grouping column by passing the column name as the 'by' parameter. Collapsing should be performed after encoding has occurred but before normalization.

### *2.2.3* Imputation and feature scaling

Most machine learning algorithms require numeric, non-missing, feature-scaled data for effective learning. With orchid-ml, one can specify strategies for imputation and scaling using the *set_normalize_options()* function which takes parameters 'nan_strat' and 'scaler_strat' to respective missing and scaling strategies. Imputation strategies include setting all unknown feature values to 0 ('zero'), or to the feature mean ('mean'), median ('median'), or most frequent ('most_frequent') values. Feature scaling is performed using a min-max scaler ('mms'), where feature values are transformed to a [0, 1] range based on the minimum and maximum values or a z-score based method ('standard'), where feature values are subtracted by their mean and divided by their unit variance. We used orchid-ml's default values for normalization, 'median' and 'standard', unless otherwise stated.

### *2.2.4* Feature selection

Classifiers with a large numbers of features can potentially begin to model noise specific to the training dataset (a.k.a. overfitting), which decreases overall performance and classification generalizability. To avoid this pitfall, we employ a feature selection method that reduces feature number to a desired subset size—generally one-tenth the number of training examples. This is accomplished through orchid-ml's *select_features()* function. This function normalizes, shuffles and divides data into training and testing sets in a 75:25 ratio. Next, it trains a user-specified model (or by default a random forest) with training data and accuracy is assessed in test data. Then, for each feature, it shuffles the feature values, remodels the data, and then compares the resulting accuracy to the original model to generate an error percentage for that feature. It repeats this process 50 times and reports mean error percentages for each feature. The specified top number of features whose permutation caused the largest decrease in model accuracy are retained for subsequent modeling.

### *2.2.5* Model generation

To model tumor data, orchid-ml first requires a label column to be set with the *set_label_column(column_name)* function. This flags one of the data columns in the *MutationMatrix* for use as class labels during supervised learning and test prediction. Orchid-ml can then perform modeling with the *svm()* or *random_forest()* function, which interface with scikit-learn's sklearn.ensemble. RandomForestClassifier and sklearn.svm.SVC modules, respectively. The Mutation Matrix() is also compatible with other sk-learn classifiers. For random forest models, we set default values of max_features='auto', max_depth = None, min_samples_split = 2 and min_samples_leaf = 1. For support vector machine models, we set the kernel default to 'linear', C = 1.0 and probability = True. Orchid-ml uses default scikit-learn values for all other parameters, but a user can pass custom sklearn parameter value pairs through orchid. To estimate model stability, orchid performs k-fold cross-validation (k = 10 by default) and reports mean accuracy and standard deviation. Optionally, it will also permute class labels, remodel data and report accuracy for comparison with a null model, which has an expected accuracy equivalent to randomly guessing a class (that is 1/C where C is the number of classes). This 'sanity check' helps ensure no systematic bias—such as large class imbalance—is falsely contributing to classification accuracy. Modeling can also be performed with custom train/test sizes by specifying the proportion of samples to withhold for testing.

#### 2.2.6 Visualization and performance

Orchid-ml includes several functions for visualizing data and reporting model performance. These functions depend on the python modules seaborn (https://seaborn.pydata.org/), matplotlib (https://matplotlib.org/) and sci-kit learn. Orchid has the ability to generate dendrograms of mutations clustered by both feature and sample in the form of the *show_dendrogram()* function and can also easily generate performance metric reports (*print_report()*); display confusion matrices (*show_confusion_matrix()*); draw violin plots to compare classification probability distributions (*show_confidence_plot()*; Supplementary Fig. S3); show Receiver Operating Characteristic (ROC) or Precision-Recall (PR) curves (*show_curves();* Supplementary Fig. S4); and indicate feature importance for classification (*show_feature_importances()*). The orchid code repository provides further documentation on each of these.

### 2.3 Application of orchid: tissue of origin dataset

We first downloaded whole-genome sequencing data from ICGC and biological annotation features as described, populating data into the multi25_20170710 database (see repository). Next, we used orchid-ml to load mutations and features, encode ordinal features and collapse mutations by patient tumor using mean feature values (this is accomplished with orchid_ml's *load_mutations()*, *load_features()*, *encode()* and *collapse()* functions respectively). This resulted in a total 960 tumor tissue profiles. From these profiles, we imputed missing data with a 'median' strategy, normalized the entire matrix with the 'mms' min/max transformation, and selected the 20 most-performant features as described. Model performance was then assessed with 10-fold cross validation and label permutation. Finally, a predictive model was generated with 65% of the data, and tissue predictions were made in the remaining 35%.

## 3 Results

### 3.1 Orchid

To facilitate the task of machine learning on tumor mutational profiles, we created orchid, an open-source software framework to efficiently annotate, manage and model tumor mutations on a genome-wide scale. A user can begin with mutational data from ICGC or in VCF format and annotation feature data in various formats, and then use orchid to import, manage, annotate and model data. Orchid is divided into two components, *orchid-db*, which loads and annotates mutations into a database system (e.g. MySQL or MemSQL), and *orchid-ml*, a python module that facilitates machine learning using the popular scikit-learn framework.

### 3.2 Orchid-db

We designed orchid-db to efficiently process, parse and transform raw data into a structured MySQL-like database to maximize subsequent access speed and analysis. Mutation and feature data can be imported individually using two orchid-db subcommands (*populate* and *annotate*), or simultaneously, in parallel, with the workflow management tool, nextflow (v. 0.17; https://www.nextflow.io/) (Di Tommaso *et al.*, 2017). For the latter option, we provide the *make_database* shell script to control nextflow execution through a single configuration file that specifies data locations and processing options. Nextflow is capable of executing seamlessly on a desktop machine, on a cluster, or in cloud environments (Amazon, DNANexus, Docker, Singularity, Apache Ignite, PBS, SGE, SLURM), and can interface with a local or remote SQL-like database system. Orchid specifically supports software compatibility with a performant, distributed, in-memory database system, MemSQL (http://www.memsql.com/) making the import of tens of millions of mutations and hundreds of feature annotations possible on the order of a few hours (see Supplementary Material for more details).

### 3.3 Orchid-ml

We also developed orchid-ml, a python module that interfaces with orchid-db data and provides convenience functions for machine learning of tumor variant data. Our module extends the pandas *DataFrame* class object into a *MutationMatrix* that adds support for importing, encoding, and subsetting tumor mutation data from the database produced by orchid-db. Our modeling functions use the scikit-learn framework for machine learning due to its flexibility, excellent documentation and large variety of algorithms. Additionally, orchid-ml is capable of visualizing data and model performance that generate plots with seaborn and matplotlib.

Once populated in the database by orchid-db, data is easily accessed and modeled with orchid-ml. A typical workflow is summarized as follows:

1. Specify access to the database generated by orchid-db with a SQL connection string.
2. Load mutations and features either in their entirety or by a desired subset (e.g. by tumor).
3. Encode categorical features using default or user-defined strategies (e.g. one-hot).
4. Optionally collapse mutations by tumor (e.g. by averaging).
5. Set a prediction label and select features.
6. Model data with any of the scikit-learn machine learning algorithms.

For convenience, random forest and support vector machine functionality is built directly into orchid-ml, automatically performing data normalization, train/test splitting, cross-validation and label permutation for null model generation. Orchid-ml visualization functions can be used to assess performance and explore relationships within the data; these include mutation dendrograms, feature weight boxplots, class prediction and confusion matrix heatmaps, receiver operating characteristic (ROC) curves and Precision Recall (PR) curves.

### 3.4 Application of orchid: tissue of origin

To demonstrate orchid's ability to facilitate machine learning with biologically relevant classification tasks, we applied a classification model used to determine tissue-of-origin from 12 tissues. The code for this task is provided as a jupyter notebook (http://jupyter.org/) in the orchid software repository.

For this application, we prepared data as described in Materials and methods and randomly sampled 100 tumor profiles for visualization with orchid-ml's *show_dendrogram()* function, using complete linkage hierarchical clustering on both features and tissues. This was done to assess segregation of tumor profiles by feature groups, and to see if patterns emerged that correspond to biology of underlying tissue type (Fig. 2).

From this we observed a small amount of tissue level grouping with particular feature combinations differentially driving segregation. For example, cancers of the stomach, uterus, head and neck and bladder appeared to show increased mutation burden in transcribed regions (Fig. 2a), and conversely lower mutation burden in repressed regions (Fig. 2e) of encode cell lines. For head and neck cancer, mutations were of higher frequency and enriched in both the

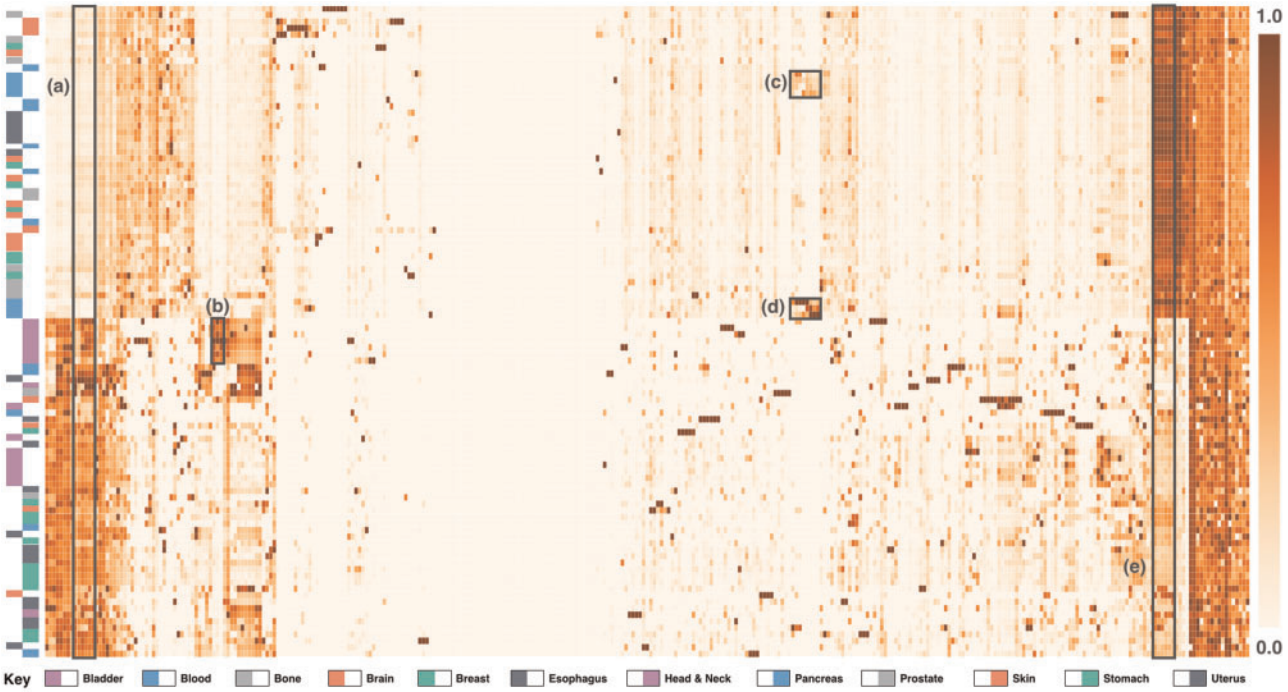


**Fig. 2.** Tumor Mutational Profile Dendrogram. Patient mutation values were averaged over all features and labeled with the tissue-of-origin. The orchid-ml *show_dendrogram()* function was then used to generate a clustered heatmap. (**a**) and (**e**) Fairly strong separation of stomach, head and neck, bladder and uterus tissues based on encode cell line transcribed regions was observed. (**b**) In head and neck cancers, a frequent, G [T>A] G context and 3′UTR mutational signal was present. (**c**) and (**d**) Blood cancers showed separation from other tissues based on the V, D, J and C immunoglobulin biotype features

G [T>A] G trinucleotide context and 3′UTR regions (Fig. 2b). Upon cross referencing the ICGC data portal, we noted the very common chr7: 140453136C [A>T] C mutation in the 3′UTR region of BRAF, representing 44% of patients, as potentially driving this signal. Finally, when considering the V, D, J and C immunoglobulin biotype features, separation of blood cancers was observed (Fig. 2c and d). We also clustered the full 960 tissues on just tissue-of-origin an observed similar patterns (Supplementary Fig. S2), as well as additional trinucleotide signatures that corresponded to those previously reported by Alexandrov *et al.* (2013).

Next, we modeled these profiles using a random forest classifier. First, we first employed feature selection to guard against overfitting by reducing the number of features from 339 to 20 using the permutation method described in Materials and methods. Of the retained features, their permutation caused an increase of between 2.5 and 5.2% in classification error. Ten of the twenty most important features were trinucleotide context features, four were transcript biotypes and two were related to cancer pathways. The remaining retained features were the modifier impact category, Nhlf enhancer, HeLa-S3 transcription and CADD. From this reduced dataset, we performed 10-fold cross validation with a random forest classifier using orchid_ml's *random_forest()* function. The resulting models had a mean accuracy for tissue classification of $0.94 \pm 0.02$. To help ensure systematic artifacts such as class label imbalance were not driving signal accuracy, orchid was used to re-train the models after permuting training labels, and the expected null performance was observed (accuracy $= 0.08 \pm 0.09$; expected $= 0.08$). Finally, the *random_forest()* function was called to build a predictive model using a randomly subset population of patients ($n = 624$; 65%), while the remaining were withheld for testing ($n = 336$; 35%). For this final model, we plotted the feature weights on a per-tissue basis (Fig. 3a), showing that several features were particularly useful for classifying just one of the tissue types (e.g. IG variable segment for blood and many of the C>T trinucleotide context features). We also used orchid-ml's *show_curves()* function to produce ROC curves in a one-vs-rest fashion for each tissue (Fig. 3b). Tissues have an AUC range between 0.80 (brain) and 0.98 (bone).

To observe whether consistent tissue misclassifications were present, we generated a confusion matrix using orchid-ml's *show_confusion_matrix()* function (Fig. 3c). For this analysis, we assigned each tumor profile the tissue with the highest predictive probability and compared the predicted tissues with their actual types. Tissues most often confused as others (False Negative Rate) include pancreas, prostate and uterus, while bone, head and neck and stomach were rarely confused. Likewise, tissues were often confused as prostate, brain and breast (False Discovery Rate), but not as often as blood, skin, uterus, esophagus. Interestingly, we also found that while some tissue types were confused in bi-directional manner (e.g. breast $\Longleftrightarrow$ prostate) others were not (e.g. pancreas $\Rightarrow$ breast).

## 4 Discussion

To better aid the analysis of tumor genomes, we present orchid, a powerful mutation management and machine learning framework. We also demonstrate orchid's ability to determine with high accuracy tissue-of-origin from tumor mutation data, which may have potential use in diagnosing tumors of unknown origin and for screening cfDNA. To our knowledge, orchid represents the first cancer mutation analysis framework with an in-memory database data storage, parallelization/cluster support and integration with python numeric analysis and machine learning modules.

While orchid does not represent the first software to annotate mutations or produce mutation profile models within machine-learning, it does offer some advantages over other methods. For one,

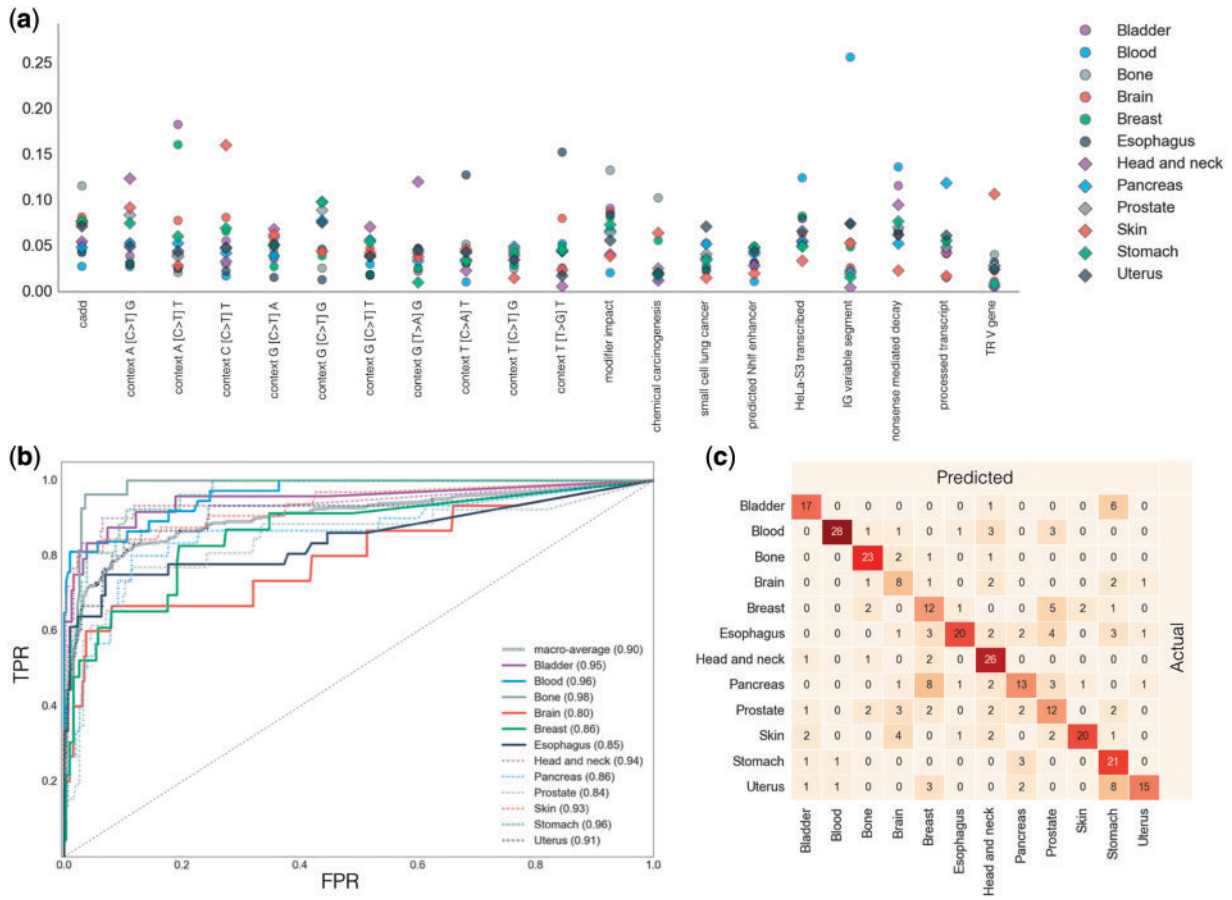


**Fig. 3.** Model Performance. (**a**) The twenty most important features for classification were selected using orchid-ml's *select_features()* function and plotted on a per-tissue basis after modeling. (**b**) The true positive versus false positive classification rates for each tissue are plotted in a one-vs-rest fashion. The dashed diagonal line indicates random classification. The macro average over all models is shown and a heavy dashed line and Area Under the Curves (AUCs) are given in parenthesis for each tissue. (**c**) A matrix indicating classification predictions from the tissue model. Rows labels are actual tissues and columns labels are tissues predicted by our model. True positive counts can be found along the diagonal, and of the remaining, false positives are along columns and false negatives are along rows

it has the ability to quickly integrate new biological features by employing a flexible parsing system with parallel processing support for both a desktop machine and cluster. Secondly, it allows seamless integration with existing python analysis workflows and provides functionality for basic machine learning tasks within the scikit-learn ecosystem. Finally, it provides many convenience functions to aid the visualization and analysis models.

Despite these advantages, orchid has a few limitations when compared to other software tools designed to model tumor mutations. For one, it's design centers around the analysis of simple somatic mutations and copy number variation data, and some cancers are largely driven by biological mechanisms of higher order genetic architecture, such as gene fusions (e.g. prostate cancer TMPRSS: ERG), large-scale structural rearrangements, epigenetic and gene expression changes. Nevertheless, the analysis of such mechanisms could be incorporated in future versions of orchid. Secondly, due to dependence on the scikit-learn ecosystem, some popular machine learning algorithms (e.g. neural networks) are not available for analysis or are not as fully featured as in other frameworks. And finally, orchid makes use of copy number variation data on a very granular mutational level, potentially missing important associations that could be seen when such data is analyzed over larger genomic regions.

With regard to our application of orchid to classify tumor tissue-of-origin, it is important to note that related methods have been previously developed. In particular, Snyder et al. used a novel nucleosome footprint window protection score to demonstrate correlation with patterns of protection and pathological states such as cancer (Snyder *et al.*, 2016). Likewise, Marquard et al. developed *TumorTracer* to classify tissue-of-origin with 85% accuracy across 6 primary sites using both somatic point mutation as well as copy number information (Marquard *et al.*, 2015). Orchid was able to achieve slightly a better accuracy of 94% among 12 tumor types, improving upon these initial methods.

While the tissue-of-origin task demonstrates one potential use of orchid, it is possible to model other types of data. For example, one can use orchid to generate a set of null, simulated mutations in conjunction with observed mutations to see if a particular feature set can be used to distinguish between the two classes, or even to assign a probability of class membership. This follows a similar strategy used by several driver/passenger and other base-level scoring tools (Fu *et al.*, 2014; Kircher *et al.*, 2014; Kumar *et al.*, 2016; Quang *et al.*, 2015) and has application in developing models for mutation prioritization for the design of custom sequencing panels for cancer detection.

## Acknowledgement

We would like to thank Daniel Himmelstein for his advice on software licensing agreements.

## Funding

This work was supported by National Institutes of Health grants CA088164 and CA201358, the UCSF Goldberg-Benioff Program in Cancer Translational Biology, Amazon Web Services and Microsoft Azure Web Services.

*Conflict of Interest*: none declared.

## References

Adzhubei,I.A. *et al.* (2010) A method and server for predicting damaging missense mutations. *Nat. Methods*, **7**, 248–249.

Agarwal,V. *et al.* (2015) Predicting effective microRNA target sites in mammalian mRNAs. *eLife*, **4**, 1–38.

Alexandrov,L.B. *et al.* (2013) Signatures of mutational processes in human cancer. *Nature*, **500**, 415–421.

Carter,H. *et al.* (2009) Cancer-specific high-throughput annotation of somatic mutations: computational prediction of driver missense mutations. *Cancer Res.*, **69**, 6660–6667.

Choi,Y. *et al.* (2012) Predicting the functional effect of amino acid substitutions and indels. *PLoS One*, **7**, e46688.

Cingolani,P. *et al.* (2012) A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff. *Fly*, **6**, 80–92.

Dees,N.D. *et al.* (2012) MuSiC: identifying mutational significance in cancer genomes. *Genome Res.*, **22**, 1589–1598.

Ernst,J. and Kellis,M. (2012) ChromHMM: automating chromatin-state discovery and characterization. *Nat. Methods*, **9**, 215–216.

Fu,Y. *et al.* (2014) FunSeq2: a framework for prioritizing noncoding regulatory variants in cancer. *Genome Biol.*, **15**, 480.

Griffiths-Jones,S. (2004) The microRNA registry. *Nucleic Acids Res.*, **32**, D109–D111.

Griffon,A. *et al.* (2015) Integrative analysis of public ChIP-Seq experiments reveals a complex multi-cell regulatory landscape. *Nucleic Acids Res.*, **43**, 1–14.

Hoffman,M.M. *et al.* (2012) Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nat. Methods*, **9**, 473–476.

Kanehisa,M. and Goto,S. (2000) KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Res.*, **28**, 27–30.

Khan,A. and Zhang,X. (2016) DbSUPER: a database of super-enhancers in mouse and human genome. *Nucleic Acids Res.*, **44**, D164–D171.

Kircher,M. *et al.* (2014) A general framework for estimating the relative pathogenicity of human genetic variants. *Nat. Genet.*, **46**, 310–315.

Kumar,P. *et al.* (2009) Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nat. Protoc.*, **4**, 1073–1081.

Kumar,R.D. *et al.* (2016) Unsupervised detection of cancer driver mutations with parsimony-guided learning. *Nat. Genet.*, **48**, 111.

Kundaje,A. *et al.* (2015) Integrative analysis of 111 reference human epigenomes. *Nature*, **518**, 317–330.

Lawrence,M.S. *et al.* (2013) Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*, **499**, 214–218.

MacArthur,J. *et al.* (2017) The new NHGRI-EBI catalog of published Genome-Wide Association Studies (GWAS Catalog). *Nucleic Acids Res.*, **45**, D896–D901.

Marquard,A.M. *et al.* (2015) TumorTracer: a method to identify the tissue of origin from the somatic mutations of a tumor specimen. *BMC Med. Genomics*, **8**, 58.

Pavlidis,N. and Pentheroudakis,G. (2012) Cancer of unknown primary site. *Lancet*, **379**, 1428–1435.

Pedregosa,F. *et al.* (2012) Scikit-Learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Pollard,K.S. *et al.* (2010) Detection of nonneutral substitution rates on mammalian phylogenies. *Genome Res.*, **20**, 110–121.

Quang,D. *et al.* (2015) DANN: a deep learning approach for annotating the pathogenicity of genetic variants. *Bioinformatics*, **31**, 761–763.

Rajagopal,N. *et al.* (2013) RFECS: a random-forest based algorithm for enhancer identification from chromatin state. *PLoS Comput. Biol.*, **9**, e1002968.

Raphael,B.J. *et al.* (2014) Identifying driver mutations in sequenced cancer genomes: computational approaches to enable precision medicine. *Genome Med.*, **6**, 5.

Snyder,M.W. *et al.* (2016) Cell-free DNA comprises an in vivo nucleosome footprint that informs its tissues-of-origin. *Cell*, **164**, 57–68.

Subramanian,A. *et al.* (2005) Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proc. Natl. Acad. Sci. USA*, **102**, 15545–15550.

Thurman,R. *et al.* (2012) The accessible chromatin landscape of the human genome. *Nature*, **489**, 75–82.

Di Tommaso,P. *et al.* (2017) Nextflow enables reproducible computational workflows. *Nat. Biotechnol.*, **35**, 316–319.

Vandin,F. *et al.* (2012) De novo discovery of mutated driver pathways in cancer. *Genome Res.*, **22**, 375–385.

Vogelstein,B. *et al.* (2013) Cancer genome landscapes. *Science*, **339**, 1546–1558.