boostDM documentation

BoostDM is a method to score all possible point mutations (single base substitutions) in cancer genes for their potential to be involved in tumorigenesis.

The method has been described and used in this study:

"In silico saturation mutagenesis of cancer genes"
Ferran Muiños, Francisco Martinez-Jimenez, Oriol Pich, Abel Gonzalez-Perez, Nuria Lopez-Bigas
DOI: 10.1038/s41586-021-03771-1
URL: https://www.nature.com/articles/s41586-021-03771-1

These docs will guide you through the comprehensive description, resources and practicalities of the method.

Latest release

This document reflects the current version v2024.07.15-cancer. For previous versions check https://boostdm.readthedocs.io and https://intogen.org/boostdm/releasednotes websites.

Resources

There are several public resources related to the boostDM framework, each tailored to a specific purpose:

Main paper

In silico saturation mutagenesis of cancer genes:
https://www.nature.com/articles/s41586-021-03771-1

Public GitHub repo containing a collection of scripts and notebooks to generate analyses and figures of the main paper: https://github.com/bbglab/boostdm-analyses

Zenodo repository providing data items generated and used in the main paper:
https://zenodo.org/record/4813082

Pipeline repository

Public GitHub repo containing the source code of the pipeline:
https://github.com/bbglab/boostdm-pipeline

Website

boostDM website: https://www.intogen.org/boostdm/ is intended for exploration of the predictions and explanations resulting from the boostDM pipeline for a collection of models meeting minimum quality criteria. The website is searchable by cancer gene, tumor type, and mutation coordinates.

Intogen

The Intogen https://www.intogen.org resource is intended for exploration of the landscape of mutations and signals of positive selection in driver genes after analyzing 33,218 tumor samples (release v2024.06.21). Intogen is instrumental for boostDM as it provides processed data that is used for the training of boostDM models.

Cancer Genome Interpreter

The Cancer Genome Interpreter https://www.cancergenomeinterpreter.org is a computational framework to interpret cancer genome variants intending to guide clinicians towards optimal decision making regarding the treatment of cancer, in particular resolving the implication of variants of unknown significance. In this context, boostDM output is used to annotate single nucleotide variants into likely drivers or non-drivers.

Introduction

To be able to identify individual cancer mutations a novel approach is required that annotates all possible mutations in a gene, independently of their probability of occurrence, as potential drivers or passengers. Instead of relying on functional impact metrics[1], this method should measure the ability of a mutation to drive tumorigenesis. Moreover, as the function of each cancer gene is different, as well as their role in tumorigenesis, we can expect that the features that define driver mutations will be different per gene.

Thus, we aim to create an approach that learns the features that define driver mutations for each cancer gene independently. In addition, the same cancer gene may have different mechanisms of tumorigenesis in different tissues (e.g. compare EGFR in Lung adenocarcinoma and Glioblastoma). Thus, if enough data is available, we aim to create gene and cancer type specific models. Furthermore, it would be desirable that such classification yields human-readable results, which help researchers point at the key features defining driver mutations in a cancer gene.

This problem has been approached before through experiments of saturation mutagenesis, in which all possible mutants of a cancer gene are generated and their impact on protein function[2–4], or cell viability[5,6] are assessed. These experiments possess obvious technical and economic hurdles. Furthermore, due to limitations imposed by the experimental setup, these approaches do not directly measure the tumorigenic potential of mutations, but rather some proxy, such as their functional impact. For instance, in certain tumor suppressor genes, saturation mutagenesis experiments have been conducted in haploid human cells to identify mutations that abrogate cell viability[5] . Only scattered mutagenesis assays have been carried out that actually assess the tumorigenic potential of mutations affecting cancer genes, restricted to few cell types, which do not represent the wide spectrum of tissue-specific constraints. Generalizing them to cover hundreds of cancer genes across cell types representing different tissues would be a herculean task.

To address this problem we have developed a platform to train machine learning models to identify driver mutations in cancer genes across cancer types. This document presents details of the methodology and its implementation.

Overview

Training set

boostDM is based on a supervised learning approach using a training set of driver and passenger mutations in cancer genes. How to define such training datasets is not obvious, as there is not a complete ground truth collection of driver and passenger point mutations in a cancer gene.

For some genes, the excess of observed-over-expected mutations is large enough that the vast majority of observed mutations are involved in tumorigenesis. We propose (and validate) that if there are enough observed mutations in cancer driver genes with consequence type above a certain excess will provide both sufficiently many mutations and high enough driver enrichment to render good discriminative ability by standard supervised classification techniques. Thus we define as positive set (drivers) the set of mutations with excess of observed-to-expected higher than 85% according to dNdScv[7].

From a theoretical perspective, passenger mutations would be any other mutations. For discriminative efficiency, however, training data must reflect the fact that passenger mutations are randomly generated following tri-nucleotide specific mutation rates (neutral mutational profile). Thus a collection of simulated mutations following the neutral mutational profile will be used as a negative set (Passengers).

Features

Each mutation provided for training is annotated with a vector of mutational features, which the classification task exploits to discriminate between observed drivers and passengers in tumors. Some mutational features of each cancer gene across malignancies have been derived from the systematic analysis of tens of thousands of tumor samples by Intogen[8] . Other relevant features have been collected from public databases: PhyloP[1], VEP[9] and PhosphositePlus[10].

Learning heuristics

For each gene-tumor type pair, the learning heuristics makes use of gradient boosted trees as the base classifier. Several base classifiers are trained on subsets (bagging) of the pool of learning mutations. Then these classifiers are aggregated into a consensus model. Alongside the training, this approach yields an out-of-bag (cross-validation) assessment of the model performance. This evaluation strategy is expected to give a conservative estimation of the performance, as it reflects the typical performance of the base classifiers before aggregation.

Scope

As a principle, boostDM provides a distinct model for each driver gene-tumor type pair. In some cases, however, the models cannot be successfully rolled out due to the fact that there are not enough mutations for training or even if there are, cross-validation yields low accuracy. Within each gene, boostDM covers the protein coding sequence of the genome with the additional inclusion of upstream/downstream 5 bps flanking each coding-sequence segment.

The effects of all mutations being considered are relative to the MANE Select transcript of protein coding genes according to the Ensembl Variant Effect Predictor version VEP.111 [9]. Note that these transcripts may include mutations in untranslated regions which are splicing affecting, even if they are non-protein-altering mutations.

Prediction

For each cancer gene and tumor type our method fits a model representing feature rules that define driver mutations in that context. Specifically, the method yields a score $0 \leq p \leq 1$ that reflects the strength of the forecast that the mutation is a potential driver: the higher the score, the stronger the evidence. Although p is not calibrated to support a probabilistic interpretation, a score > 0.5 reflects predominant evidence in favor of the mutation being a potential driver.

Explanations

Each base classifier (gradient boosted trees) admits an additive explanation model that decomposes the logit prediction of each individual mutation as the sum of so-called SHAP values associated with the features[11]. These are explanatory values in the sense that a feature having positive (resp. negative) SHAP value implies that the method deems more likely (resp. unlikely) that the mutation is a driver conditioned to the feature value. We define the SHAP values of the aggregated model as the mean SHAP values across base classifiers.

Pipeline

Input

The full output of Intogen v2024.06.21 is required to conduct the preprocessing steps of the boostDM pipeline. As per Intogen version v2023.05.31 the data required by boostDM is streamlined in a specific format as part of the Intogen output.

There are two main folders, referred to as INTOGEN_DATASETS and BOOSTDM_DATASETS, generated by Intogen which boostDM feeds on. BOOSTDM_DATASETS is generated during the installation process of Intogen, whereas INTOGEN_DATASETS is generated as an output of Intogen upon running the pipeline. Please, check out the Intogen documentation: https://intogen-plus.readthedocs.io/en/latest/index.html.

Steps of the pipeline

The pipeline is run with Nextflow DSL=1 and it has been tested with Nextflow version 20.07.1. The pipeline is divided into six steps that are run separately with Nextflow scripts.

```
01_training.nf
02_discovery.nf
03_model-selection.nf
04_prediction.nf
05_output_plots.nf
06_benchmarks.nf
```

To run each Nextflow script, the following command has to be executed:

```
nextflow run <nexflow_script>.nf -resume -profile <profile>
```

The first four steps are the main steps of the pipeline, from model training through to model selection and in silico saturation mutagenesis. Steps 5 and 6 can be considered supplementary steps required to generate the basic graphical representations of the output and benchmarks against experimental mutational scanning data and other bioinformatic prediction methods.

Data dependencies: nextflow.config

To run the pipeline it is necessary to specify to Nextflow the paths of the data dependencies. We specify those dependencies as environment variables using the following clause in the nextflow.config file:

```
env {  GENOME_BUILD = "hg38"
       INTOGEN_DATASETS = "./intogen_datasets/"
       BOOSTDM_DATASETS = "./boostdm_datasets/"
```

```
        VEP_SATURATION = env.INTOGEN_DATASETS + "/steps/boostDM/saturation/"
        PIPELINE = "./boostdm-pipeline-2024/"
        OUTPUT= "./boostdm-output/"
        MAVE_DATA = "./mave_data/"}
```

The only dependency that is not provided by Intogen is the MAVE_DATA folder. This is a manually curated resource where different catalogs of mutations are annotated with experimental saturation mutagenesis readouts (multiplexed assays of variant effects or MAVE). The current boostDM release has been run with the v2024.08.16 release, which can be forked from github: https://github.com/bbglab/mave-data/releases/tag/v2024.08.16.

In the current release the benchmark analysis includes data whereby the fitness and/or functional impact of mutations at TP53, PTEN and Ras genes has been put to test. Source data has been retrieved from the following studies:

- Giacomelli et al.[12] (TP53) DOI: 10.1038/s41588-018-0204-y
- Kato et al.[3] (TP53) DOI: 10.1073/pnas.1431692100
- Kotler et al.[13] (TP53 DNA-binding domain) DOI: 10.1016/j.molcel.2018.06.012
- Ursu et al.[14] (TP53, KRAS) DOI: 10.1038/s41587-021-01160-7
- Boettcher et al.[15] (TP53) DOI: DOI: 10.1126/science.aax3649
- Mighell et al.[6] (PTEN) DOI: 10.1016/j.ajhg.2018.03.018
- Matreyek et al.[16] (PTEN) DOI: 10.1038/s41588-018-0122-z
- Bandaru et al.[17] (Ras domain) DOI: 10.7554/eLife.27810

Software dependencies

The software dependencies of the pipeline are handled with containers: the pipeline runs on an interpreter that already fulfills all the software dependencies.

Two containers are needed, **boostdm.simg** and **ensembl-vep_111.0.sif**, which must be specified in the nextflow.config file:

```
singularity {
    enabled = true
    cacheDir = "./singularity_images/"
    runOptions = "-B " + env.PIPELINE + "/containers_build:/boostdm"
}

process {
    cpus = 1
    executor = 'slurm'
    queue = 'normal,bigrun'
    errorStrategy = 'ignore'
    withLabel: boostdm {container = "file:///${singularity.cacheDir}/boostdm.simg"}
    withLabel: vep {container = "./ensembl-vep_111.0.sif"}
}
```

Build

boostdm.simg

Built from the recipe provided in the boostDM repo:
`./boostdm-pipeline/containers_build/boostdm/Singularity`

ensembl-vep_111.0.sif

It can be pulled from https://hub.docker.com/r/ensemblorg with the following command line:

```
singularity pull --name vep.sif docker://ensemblorg/ensembl-vep:release_111.0
```

Pipeline output

The pipeline output is provided in the following folder tree structure:

```
├── benchmarks
│   ├── cv_tables
│   ├── cv_tables_annotated
│   ├── cv_tables_annotated_chasmplus
│   ├── pr_plots
│   ├── saturation_dbNSFP
│   ├── saturation_mave
│   ├── vep_input
│   └── vep_output_dbNSFP
├── create_datasets
│   └── <intogen cohort>.regression_data.tsv
├── discovery
│   └── discovery.tsv.gz
├── evaluation
│   └── <tumor types>
│       └── <gene>.eval.pickle.gz
├── features_group
├── model_selection
├── output_plots
│   ├── blueprints
│   ├── clustered_blueprints
│   └── discovery_bending
├── saturation
│   ├── annotation
│   │   └── <gene>.<tumor type>.annotated.tsv.gz
│   └── prediction
│       └── <gene>.model.<ttype model>.features.<ttype features>.prediction.tsv.gz
├── splitcv
│   └── <intogen cohort>.cvdata.pickle.gz
├── splitcv_meta
│   └── <tumor types>
│        └── <gene>.cvdata.pickle.gz
└── training_meta
        └── <tumor types>
             └── <gene>.models.pickle.gz
```

Hereto we describe the main outputs in detail.

Preprocessed regression data

`create_datasets/<intogen cohort>.regression_data.tsv`

This is a collection of positive and negative training mutations alongside their features for a given cohort. The columns are:

alt: alternate allele, chr: chromosome, gene: gene, pos: position in genomic coordinates (hg38), ref: reference allele, PhyloP: PhyloP conservation score, aachange: amino acid change, nmd: NMD annotation; Acetylation, Methylation, Phosphorylation, Regulatory_Site, Ubiquitination: post-translational modification annotation, CLUSTL: oncodriveCLUSTL ordinal value in {0,1,2}, motif: Pfam domain(s) that overlap the mutation; csqn_type_missense, csqn_type_nonsense, csqn_type_splicing, csqn_type_synonymous: simplified protein coding consequence type, HotMaps: HotMAPs ordinal value in {0,1,2}, smRegions: smRegions ordinal value in {0,1,2}, response: positive/negative driver label for supervised learning.

Data splits for training

`splitcv/<intogen cohort>.cvdata.pickle.gz`

Pickled dictionaries indexed by the set of driver genes in the cohort. For each gene the dictionary value is a list of tuples (as many elements as 50 base models), each tuple consisting of four dataframes. Each tuple represents a train-test split:

- df_covariates_train (pandas.DataFrame, feature information)
- df_covariates_test (pandas.DataFrame, feature information)
- df_response_train (pandas.Series, response labels, binary)
- df_response_test (pandas.Series, response labels, binary)

The columns are:

chr, pos, ref, alt, CLUSTL, HotMaps, smRegions, PhyloP, nmd, Acetylation, Methylation, Phosphorylation, Regulatory_Site, Ubiquitination, csqn_type_missense, csqn_type_nonsense, csqn_type_splicing, csqn_type_synonymous (see description in the previous section).

`splitcv_meta/<tumor type>/<gene>.cvdata.pickle.gz`

Pickled list of tuples (as many elements as base models) of four elements following the same structure as described above. In this case, the splits are the result of grouping by the splits by gene and tumor type.

Saturation annotation files

`saturation/annotation/<gene>.<tumor type>.annotated.tsv.gz`

Table containing a comprehensive catalog of all possible mutations mapping to the MANE Select transcript region of interest according to the results of Intogen plus annotations from VEP, PhyloP and PhosphositePlus.

The columns are:

chr, pos, Reference, alt, ENSEMBL_GENE, ENSEMBL_TRANSCRIPT, Feature_type, cDNA_position, CDS_position, Protein_position, Amino_acids, Codons, Existing_variation, Impact, Distance, Strand, Flags, gene, Symbol, source, HGNC_ID, CANONICAL, MANE_SELECT, Mane_plus_clinical, ENSP, PhyloP, aachange, nmd, Acetylation, Methylation, Phosphorylation, Regulatory_Site, Ubiquitination, CLUSTL, HotMaps, motif, smRegions, csqn_type_missense, csqn_type_nonsense, csqn_type_synonymous, csqn_type_splicing.

Discovery index output

`discovery/discovery.tsv.gz`

Discovery index output summary in tabular format. The columns are:

gene, ttype: tumor type, n_muts: number of observed mutations in gene in tumors matching the tumor type, n_unique_muts: among them, number of unique mutations, n_samples: total number of samples matching tumor type with at least one mutation in gene, discovery_index: median discovery index estimate for the gene and tumor type, discovery_high: upper confidence bound (IQR) for the discovery index, discovery_low: lower confidence bound (IQR) for the discovery index.

Gradient boosting base classifiers

`training_meta/<tumor type>/<gene>.models.pickle.gz`

The set of trained base models for a given gene and tumor type are kept as pickled dictionaries with the following keys:

models: list of trained base classifiers, boosted tree model instances (boostwrap.methods.Classifier)

x_test: list of pandas.DataFrame instances with feature information used for testing at each base classifier

y_test: list of pandas.Series instances with response labels used for testing at each base classifier learning_curves: list of dictionaries with performance evaluation vectors at train and test (validation_0 and validation_1), useful for representation of the learning curves depicting the learning progress at each step of the training (number of estimators) for each base classifier.

Performance summary of base classifiers

`evaluation/<tumor type>/<gene>.eval.pickle.gz`

The information for the base classifiers trained for a given gene in a given tumor type. This is a dictionary with correlative lists of values for each of the following metrics:

ROC-AUC score (auc): under the ROC curve, MCC score (mcc): Matthews correlation coefficient, Log-loss (logloss): cross-entropy, Precision (precision): also known as positive predictive value, true-over-positive rate; Negative predictive value (npv): false over negative rate, Recall (recall): also known as sensitivity, positive over true rate; F-score (fscore): harmonic mean between precision and recall; F-score50 (fscore50): weighted F-score priming precision over recall; Accuracy (accuracy): rate of correctly classified mutations, Test dataset balance (balance): automatically check of balance of labels, perfect balance should yield 0; Test size (size): total number of mutations in the test set; Calibration (calibration): The relative difference between the average boostDM scores and the average positive labels, expressed as

$$C = \frac{(mean(\hat{y}) - mean(y))}{mean(y)}$$

In the case of perfect calibration, it should give values C~0.

In silico saturation mutagenesis predictions

```
saturation/prediction/
<gene>.model.<ttype model>.features.<ttype features>.prediction.tsv.gz
```

Predictions for all possible mutations in the MANE Select transcript region of interest for a specific gene in a tumor type. The columns are:

gene, ENSEMBL_TRANSCRIPT, ENSEMBL_GENE, chr, pos, alt, aachange, CLUSTL, HotMaps, smRegions, PhyloP, nmd, Acetylation, Methylation, Phosphorylation, Regulatory_Site, Ubiquitination, csqn_type_missense, csqn_type_nonsense, csqn_type_splicing, csqn_type_synonymous, selected_model_ttype, boostDM_score, boostDM_class, shap_CLUSTL, shap_HotMaps, shap_smRegions, shap_PhyloP, shap_nmd, shap_Acetylation, shap_Methylation, shap_Phosphorylation, shap_Regulatory_Site, shap_Ubiquitination, shap_csqn_type_missense, shap_csqn_type_nonsense, shap_csqn_type_splicing, shap_csqn_type_synonymous.

The columns with “shap_” prefix denote the SHAP values corresponding to the prefixed features. “boostDM_score” and “boostDM_class” denote the prediction score and whether this score is higher than 0.5 which the method established as the threshold for driver potential.

Full description

Implementation

Each boostDM model is an ensemble of base classifiers (n=50) each trained with a random sample subset (bagging) drawn from the learning dataset. Each base classifier is a boosted trees model (sum of regression tree functions trained with XGBoost gradient boosting implementation) with a cross-entropy loss function. The base classifiers are aggregated into a consensus model with an aggregator intended to correct for the systematic bias of each expert classifier (Section~\ref{consensus}). The consensus model additive explanations are computed as the mean SHAP values base classifiers (Section~\ref{explanations}).

The boostDM pipeline has been implemented in Python and Nextflow[18]. The models were implemented with the libraries xgboost[19] (version 0.90) to train the base classifiers and shap[11] (version 0.28.5) to compute the SHAP values associated with the predictions.

Base classifier hyperparameters

The model hyperparameters specify the learning strategy to keep a balance between loss minimization and generalization. While some hyperparameter values are the result of an explicit design decision, others cannot be unequivocally defined. The current values stand as a compromise resulting from a testing series.

Herein we provide the current hyperparameter set-up:

- Models are sums regression tree functions functions (booster="gbtree")
- The learning task minimizes a cross-entropy loss function (objective= "binary:logistic").
- All the features are available when building each new tree (colsample_bytree=1) and each new tree level (colsample_bylevel=1).
- Learning rate (learning_rate=0.001) was decided on by testing in combination with the maximum number of training steps.
- Percentage of samples randomly drawn prior to growing a new tree is 70% at every iteration (subsample=0.7). This is a technical choice to further prevent overfitting.
- Maximum depth of trees used in the tree function (max_depth=4). In practice a good performance can be attained even at depth as low as 1 (stumps). However, at a higher but still tolerable computational cost, more depth gives more room for learning feature interactions.
- Default regularization hyperparameters were employed.
- Maximum number of training steps (n_estimators = 20000).

In view of performance tests, expected capacity for the models to handle feature interactions and moderate training runtime, we deemed this a convenient choice. Notwithstanding the importance of this setup, due to the fact that our base classifiers undergo additional aggregation, biases attributable to gradient boosting training will tend to moderate, resulting in an implicit form of regularization.

Tumor type ontology

A specific tumor type determines the set of sequenced samples that contribute positive mutations to the learning dataset, the neutral mutation profile required to simulate passenger mutations and the value of some features.

We aim to develop models that are capable of classifying mutations across tumor types with different degrees of generality, the main reason being that for some specific gene-tumor type pairs the number of mutations is not sufficient to render a good model fitting, but the lack of mutations to train a gene-tumor type model can potentially be mitigated by pooling mutations from other samples of similar tumor types from a histopathological perspective.

We delineate a model selection strategy based on a hierarchical organization of tumor types that allows to conduct this pooling in a systematic way. Thus we defined a tumor type ontology (oncotree) employed in Intogen [8] that allows us to group samples according to several degrees of specificity regarding the tissue-pathology context where the mutations are reported. Then the root term CANCER is connected to two children terms, SOLID and NON_SOLID, which are connected to respective children at increasing levels of specificity.

As tumor type ontology for hierarchical model implementation across tumor types we now use the oncotree version boostdm2023 which is based on the 2021 version of the MSKCC oncotree (https://oncotree.mskcc.org).

Each model trained by boostDM is relative to a gene and tumor-type pair, where the tumor type corresponds to a term in the Oncotree: we denote it as (G,T)-model for brevity.

Filtering

Consequence type

We restricted our analysis to single nucleotide substitutions annotated with Sequence Ontology[20] terms (according to VEP.111) with respect to the region of interest (coding sequences including 5bp flanks as per MANE Select). The pipeline contemplates four main consequence types of interest: “missense”, “nonsense”, “splicing” and “synonymous”.

- The term “synonymous” is equivalent to the SO term synonymous_variant unless the variant is also annotated by VEP.111 with another consequence type that is more deleterious.
- The term “missense” is equivalent to the SO term missense_variant.
- The term “nonsense” includes the SO terms:

stop_gained
stop_lost
start_lost

- The term “splicing” now includes the SO terms:
  splice_donor_variant
  splice_acceptor_variant
  splice_region_variant
  splice_donor_5th_base_variant
  splice_donor_region_variant
  splice_polypyrimidine_tract_variant
  intron_variant

Multiple nucleotide variants

Adjacent point mutations were excluded from our analysis due to the plausible risk that these are misannotated.

Mutational features

Given a point mutation in a driver gene and tumor type (either observed or randomized) our method requires an annotation of the mutation with a set of features to train the classifiers.

Consequence type

Every mutation was annotated with the following dummy-encoded features: “missense”, “nonsense”, “synonymous” and “splicing”. For all observed and synthetic mutations in driver genes, we retrieved the annotations from VEP.111 for the MANE Select transcript.

Clustering features

By clustering features we mean the dichotomous annotation indicating whether the mutation maps a significant linear cluster, 3D cluster and Pfam domains (significantly deviating from the background mutation rate model). These are the only mutational features employed by boostDM that are specific to the tumor type context.

These features are inferred by Intogen at the cohort level, i.e. for each Intogen cohort the methods oncodriveCLUSTL (linear clusters), HotMAPS (3D clusters) and smRegions (enriched Pfam domains) cast site specific annotations indicating whether the site maps a significant cluster according to the method’s criterion.

For boostDM training and prediction purposes, given a clustering method and a query mutation in a tumor type context, we deem that the mutation maps to a cluster in that tumor type according to the method if it maps to a cluster according to the method in any Intogen cohort that belongs to the tumor type (see Oncotree section).

Linear clusters identified by oncdriveCLUSTL

For every mutation we annotated using a 3-level ordinal feature value whether it overlaps a significant linear cluster identified by the method OncodriveCLUSTL[21] in the same tumor type (level=2), in a different tumor type (level=1) or it does not overlap any significant linear cluster (level=0).

3D clusters identified by HotMAPS

For every mutation we annotated using a 3-level ordinal feature value whether it overlaps a significant 3D cluster identified by the method HotMAPS[22] in the same tumor type (level=2), in a different tumor type (level=1) or it does not overlap any significant 3D cluster (level=0).

Significantly enriched Pfam domains identified by smRegions

The method smRegions[23] identifies Pfam domains[24] that are significantly enriched for mutations in a gene across tumors of the cancer type. For every mutation we annotated using a 3-level ordinal feature value whether it overlaps a significant Pfam domain identified by the method smRegions in the same tumor type (level=2), in a different tumor type (level=1) or it does not overlap any significant Pfam domain (level=0).

Phylogenetic conservation across vertebrates

Conservation of the reference nucleotide across vertebrates measured through the PhyloP 100-way score[1].

Post-translational modifications

Non-synonymous coding mutations were also annotated with post translational modifications (PTMs) when the mutation affected an amino acid that is known to be acetylated, phosphorylated, ubiquitinated, methylated or subject to any other regulatory modification according to PhosphositePlus [10].

NMD

Whether nonsense mutations overlap the first or last coding exon of the transcript (according to VEP.111) reflecting the potential for the truncating variant to skip nonsense-mediated mRNA decay[25,26].

Excess of mutations

The so-called excess of mutations for a given coding consequence-type quantifies the proportion of observed mutations at this consequence-type that are not explained by the neutral mutation rate. The excess is computed from the dN/dS estimate ω computed by dNdScv as $(\omega - 1)/\omega$. We computed the excess for missense, nonsense and splicing-affecting mutations.

Training

Training datasets

The first requirement for our supervised learning approach is to create a catalog of mutations labeled as Drivers or Passengers. This catalog is established globally for all the models, then for the training of each (G,T)-model only the mutations relevant to the (G,T) context are used.

Drivers

The positive set (herein “Drivers”) used for training are observed mutations in mutational cancer genes (Intogen) that exhibit a consequence-type specific excess of observed-over-expected mutations higher than 85% according to the method dNdScv[7]. Repeated mutations (i.e., when the same mutation is observed in different samples) are allowed in the set of Drivers.

Passengers

For each Driver mutation mapping to a gene and cohort of samples, we randomly select one mutation from the region of interest following probabilities the tri-nucleotide specific probabilities recorded in the cohort. Because we are bound to train 50 base classifiers, we do this random selection 50 times with replacement.

Data Splits

For each gene-tumor type pair (G,T), the corresponding base classifiers are obtained after conducting training with two sets of annotated mutations, namely Train and Test. We will refer to a Train-Test pair as a Split. In our setting each Split is generated randomly and must satisfy the following requirements:

- Both Train and Test sets are Driver-Passenger balanced.
- The Train set comprises 70% of Driver examples.
- Each unique Driver instance belongs either to Train or Test.
- Repeated Driver mutations (i.e., when the same mutation is observed in different samples) are allowed in Train, but not in Test (to prevent spurious inflation of the cross-validation performance evaluation).
- Passenger mutations in Train and Test are randomly selected among all the mutations in the Passengers pool matching the (G,T) context.

We set the minimum average number of mutation instances in the Train size to deem the models trainable at 30.

Cross-validation and early stopping

When training a base classifier, we implemented sequential evaluation to determine the optimal number of estimators. We evaluate the performance after each learning step using the Test dataset. This sequential evaluation informs whether the training must stop due to steady or decreasing performance for a set of consecutive iterations (early stopping).

We use the cross-entropy as a performance measure to assess training progress sequentially with cross validation.
See e.g.: https://scikit-learn.org/stable/modules/model_evaluation.html#log-loss

We define an early-stopping requirement of 2,000 iterations. Thus each expert classifier training finishes when either of the following conditions is first met: i) the maximum number of training steps (N=20,000 iterations) is attained; ii) the model's performance on the Test data set does not improve for 2,000 consecutive iterations.

After the training we record the performance of the classifier via log-loss and a weighted F-score (F-score50) when applied to the Test dataset. Although cross-validation and early stopping allow some extent of data leakage from the Test dataset, it turns out to provide a conservative performance evaluation compared to real performance in
sample-wise hold-out tests.

Shapley Additive Explanations

Each base classifier yields an additive explanation model based on Shapley Additive Explanations [11] (SHAP). Specifically, each expert classifier can decompose the logit forecast yielded by a particular mutation m, say logit p(m), into a collection of SHAP values $s_i(m)$, one per feature, in such a way that $s_1(m)\ +...+\ s_n(m)\ =\ logit\ p(m)$ for all m. The intuition behind the concept of SHAP value is rooted and better understood in the context of cooperative game theory, where the Shapley value was originally described[27]. If we think of logit p(m) as a utility function that depends on the contribution of a coalition of features (thought of as players of a cooperative game) the SHAP values account for the relative contributions of the features by averaging the differences between: i) the expected forecast when the feature value is known and ii) the expected forecast when the feature value is not known.

Aggregator of base classifiers

One challenge of our approach is that during training a fraction of the Driver labels may correspond to passenger mutations, even if this fraction will in general be lower than 15% (recall that we only consider those mutations matching a consequence-type specific dNdScv excess higher than 85%). This implies that each individual expert classifier may be at risk of being biased by the specific choices of Drivers made in the Splits. To work around this difficulty, we propose to use a pool of base classifiers, each trained with a different partial view of the data, so that the final forecast and explanation results from combining these classifiers, while also correcting for the systematic biases that they may bear.

Forecast aggregator

Our approach resorts to a non-linear combination of probabilities based on a logit-normal model[28]. Specifically, if a base classifier B_i casts a prediction p_i that a specific mutation is a driver and $Y_i = logit\,(p_i)$, this modeling choice assumes that p_i arises from some latent true probability p that the mutation is a driver, which has in turn been moderated by some degree of systematic bias $Y_i = 1/a \cdot logit(p) + \epsilon_i$, with $\epsilon_i \sim N(0, \sigma^2)$ being a normal random variable with standard deviation σ and a>0 representing a systematic bias. Intuitively, the systematic bias quantifies the extent to which individual classifiers regress towards a log-odds zero due to partial information or under-confidence while issuing their respective forecasts. While a=1 would be associated with an unbiased forecast, a>1 represents under-confidence.

Using the previous modeling assumptions, given a collection of classifiers and their respective forecasts, the latent true probability p can be estimated with the following estimator:

$$\hat{p} = \frac{exp(a\overline{Y})}{1+exp(a\overline{Y})}$$

where $\overline{Y}$ denotes the average of the logits a>1 is the (underconfident) systematic bias.

The level of systematic bias gives us a method to sharpen the consensus forecasts of a coalition of underconfident predictors. In the interest of clarity and interpretability of the outcome, we require boostDM forecasts to have a marked bimodal tendency to yield forecasts close to either 0 or 1. Upon testing the method in some exemplary gene-tumor type contexts, we committed to a uniform choice of a=2.3 for all models.

SHAP aggregator

The SHAP vector associated to a specific input mutation is computed as the feature-wise mean of the SHAP vectors cast across base classifiers.

Model Selection

As a principle, for each (G,T) context we can compute a classification model as an aggregator of base classifiers. A weak predictive power can come about either because there are not enough observed mutations to render a clear signal by means of regression with the current features, or because regardless of the number of mutations the set of mutational features is not sufficient to learn a distinct signal for the Drivers. A weak cross-validation predictive ability is a clear sign that the model is up to the learning problem, however a strong performance is not an unequivocal sign that the model is solving the task properly. Herein we discuss the criteria to deem a model sufficient and the rule to decide which (general) model to use given a target gene-tumor type pair.

Evaluation in test sets

Train-test splits give rise to base classifiers that are ultimately merged into a consensus classifier. These splits allow performance assessment by casting the predictions of the base classifiers trained with the split on the test set (cross-validation). The performance measure used to evaluate each partial classifier is a weighted F-score (F-score50). The choice of the weighted F-score is motivated by the fact that some (up to 15% given one of the thresholds set for building models) “driver” mutations may actually be passengers (see below). From the 50 splits computed to generate the consensus classifier a global performance summary is derived taking the mean F-score50 across base classifiers.

To establish a simple performance criterion we resort to a weighted F-score that gives more importance to precision than to recall. We used an F-beta score with beta=0.5 as defined in https://en.wikipedia.org/wiki/F-score, which would measure the effectiveness of retrieval from the perspective of a user who attaches twice as much importance to precision as recall. We denote this as F-score50 throughout.

First, in assessing driverness, false negatives are in general more tolerable than false positives for a classifier with practical implications, as the real set of drivers is generally assumed to be much smaller than the set of all possible mutations that can be generated under neutral selection. Second, due to the intrinsically impure set of the mutations used for learning, some driver mutations in the test set are expected not to be true driver mutations. Consequently, a theoretically perfect classifier could not even attain perfect recall, hence recall must be undervalued in order to assess the method’s performance.

Representativeness

Intuitively, the performance criterion delineated in the previous section is not sufficient to establish the confidence of the assessment. For example, low confidence would follow if the repertoire of observed mutations upon which the model is trained are not representative of the set of potentially observable mutations in the gene-tumor type context. Therefore, the model selection criteria must be endowed with a measure of representativeness of the observed mutation set.

To measure representativeness we make use of two properties: i) the propensity to observe the same mutations on repeated subsamples (measured with the discovery index) and ii) the number of mutations observed. We implement a simple rule that requires more observed mutations if the discovery index is lower.

Model selection

Thus, our model selection strategy can be thought of as conducted in two steps:

Step 1: models are only trained if there are at least 30 mutations on average across the training splits of the 50 base classifiers. Note that the training splits are label-balanced sets comprising

70% of the total mutation training set. Note that the size of the mutation set available for training does in turn depend on the dN/dS excess per consequence type inferred at each cohort.

Step 2: we apply a composite rule that requires:

- mean CV performance F-score50 across base classifiers >= 0.8
- a minimum number of observed mutations per each tier defined by the discovery index

Step 2 reads in Python code as follows:

```
DISCOVERY = (0, 0.5, 0.75)
MUTATION = (50, 30, 0)
FSCORE = 0.8

def meet_condition(fscore, discovery, n_muts):
     if fscore >= FSCORE:
          for d, n in zip(DISCOVERY, MUTATION):
               if (discovery >= d) and (n_muts >=n):
                    return True
     return False
```

In silico saturation mutagenesis

For all gene and tumor type combinations, we can annotate all the possible mutations (SNVs) in the MANE Select region with the mutational features as described. With these annotations and the trained boostDM models we can in principle cast the predictions of boostDM for all mutations. We could potentially do this exercise with all available models for the given gene, but this would make little sense, for some model contexts would bear little resemblance with the context of the query mutations.

For each gene and tumor type context for which we have mutation annotations, we cast the prediction with all models for the given gene as long as the following requirements are met:

- The model has been trained in a tumor type context that is the same or more general than the tumor type context of the query mutations.
- The model meets the model selection criteria described above.

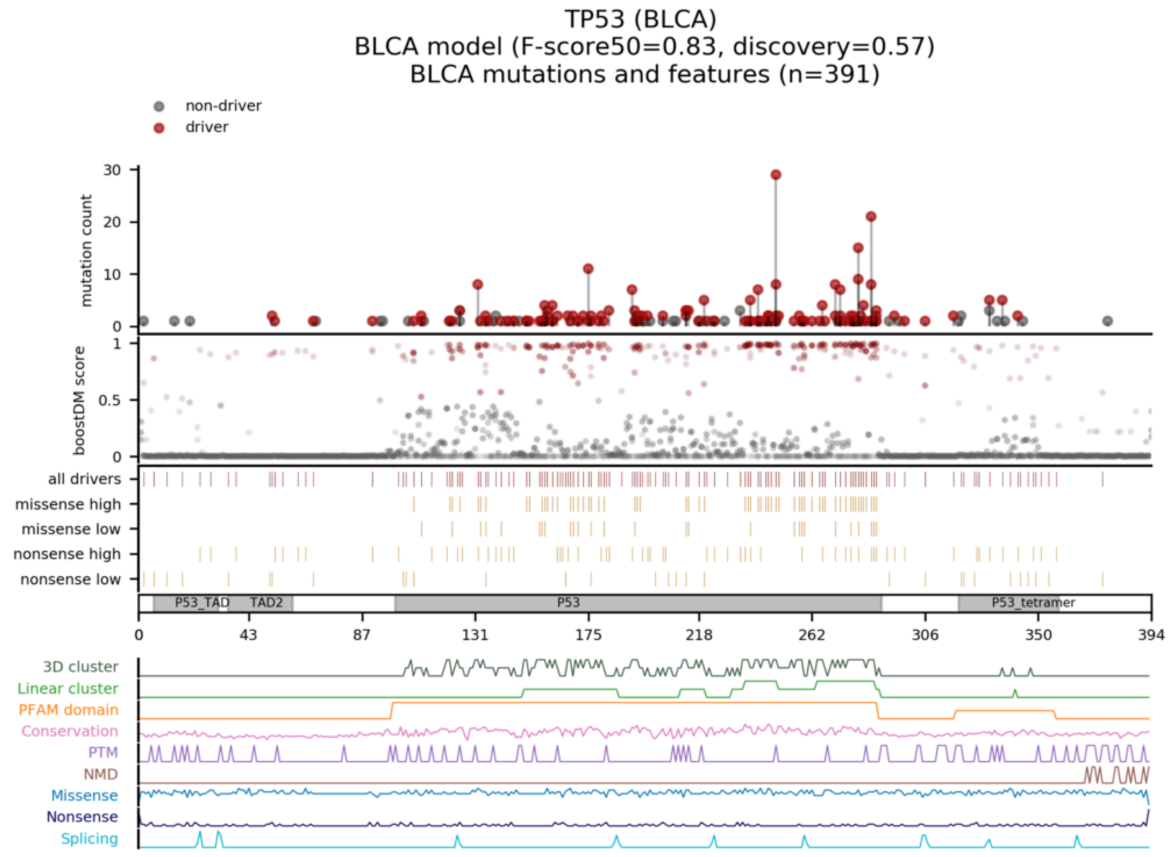
As a result, we provide prediction files for 878 gene/model tumor type/feature tumor type combinations. These files encompass the feature values, boostDM predictions and local feature explanations (SHAP) for all the possible SNVs in the MANE Select region of interest.

Output plots

We briefly describe the three main visualizations that we have included in the pipeline. These plots are used in the boostDM website.

Blueprints

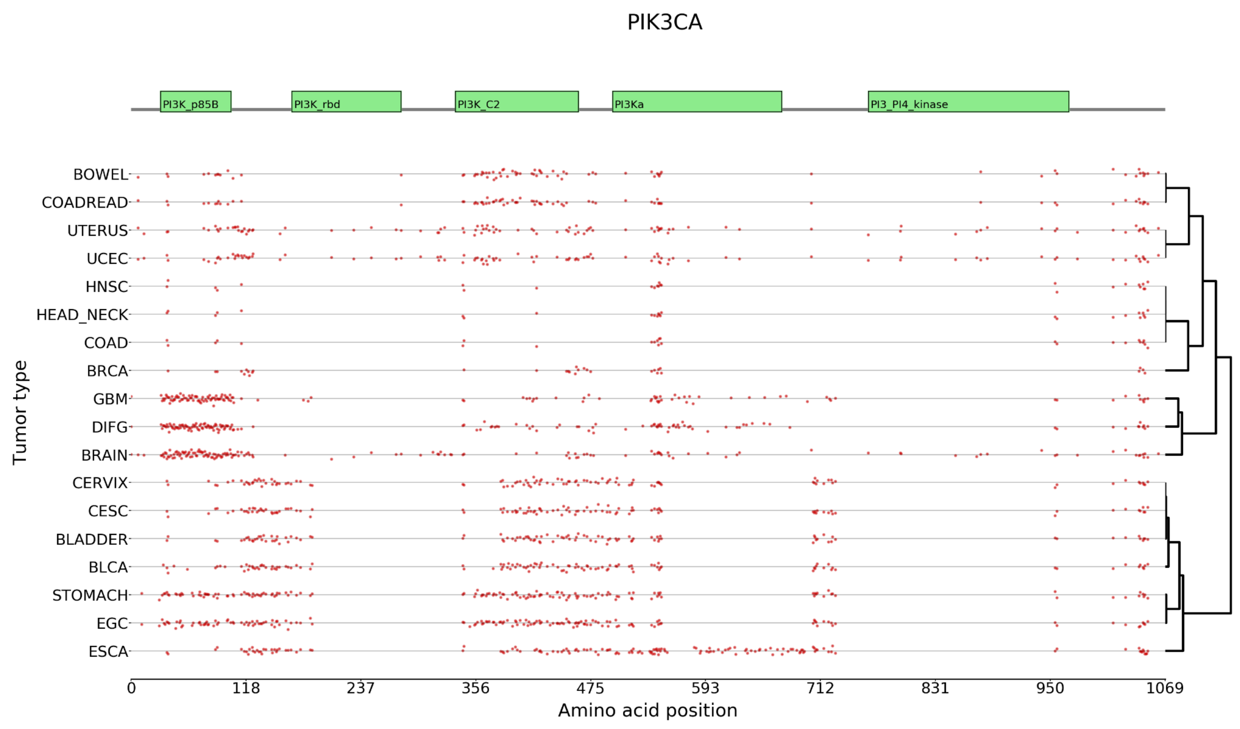
Stack representation mapping the gene protein coordinates with different annotation layers, from top to bottom: observed mutation frequencies, boostDM score, missense/nonsense driver mutations with high and low confidence tiers, Pfam domains, feature annotations.



Clustered blueprints

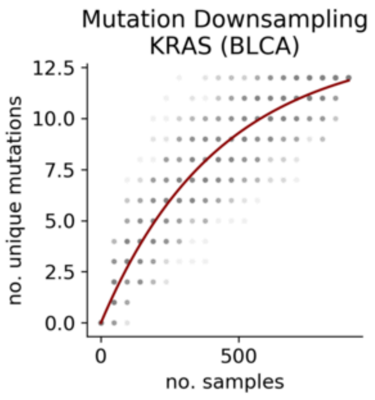
Hierarchical clustering representation of several in silico saturation mutagenesis predictions across the collection of tumor types for which we can cast a prediction with the same model tumor type model as features tumor type, i.e. there is a good quality model for the gene in the tumor type context and this model is used to cast the predictions feeding on the feature annotations calculated from this tumor type context.

Each red dot represents an amino acid position where at least one SNV is deemed a driver by boostDM. The hierarchical clustering is based on the MCC similarity between the amino acid binary vectors encoding the driver/non-driver prediction.



Discovery index

Scatter plot and best inverse exponential fitting curve representing the subsampling experiment used in the calculation of the discovery index. The gray dots represent the number of unique mutations in the pool of KRAS mutations of random subsets of tumors taken from the whole tumor type cohort. The inverse exponential fitting bending indicates how close to saturation, i.e. the more the curve is bent, the less likely new mutations are discovered as we sequence more.



Benchmarks

Given the importance for post-hoc evaluation, the boostDM pipeline now includes some benchmarks in the same fashion as those accompanying the first release of the method (https://doi.org/10.1038/s41586-021-03771-1). We have included both experimental saturation mutagenesis datasets and bioinformatic scores retrieved through dbNSFP (https://sites.google.com/site/jpopgen/dbNSFP).

Datasets

Experimental saturation mutagenesis datasets

We collected experimental saturation mutagenesis assays whereby the fitness and/or functional impact of mutations at certain human genes has been put to test in highly multiplexed experimental setups.

- Giacomelli et al. (TP53): https://doi.org/10.1038/s41588-018-0204-y
- Kato et al. (TP53): https://doi.org/10.1073/pnas.1431692100
- Kotler et al. (TP53 DNA-binding domain): https://doi.org/10.1016/j.molcel.2018.06.012
- Ursu et al. (TP53, KRAS): https://doi.org/10.1038/s41587-021-01160-7
- Boettcher et al. (TP53): https://doi.org/10.1126/science.aax3649
- Mighell et al. (PTEN): https://doi.org/10.1016/j.ajhg.2018.03.018
- Matreyek et al. (PTEN): https://doi.org/10.1038/s41588-018-0122-z
- Bandaru et al. (Ras domain): https://doi.org/10.7554/eLife.27810

Bioinformatic scores

We have also taken advantage of a collection of bioinformatic scores aiming to measure the pathogenicity or functional impact of variants for comparison against boostDM. Although most of these tools were not designed specifically to measure the likelihood of the variants to be under positive selection in the context of somatic evolution, all these scores are relevant and have been used in cancer genomics related studies and methods.

- CADD: https://doi.org/10.1093/nar/gky1016
- FATHMM: https://doi.org/10.1093/bioinformatics/btv009
- MetaLR: https://doi.org/10.1093/hmg/ddu733
- MetaRNN: https://doi.org/10.1101/2021.04.09.438706
- MutationAssessor: https://doi.org/10.1093/nar/gkr407
- PROVEAN: https://doi.org/10.1093/bioinformatics/btv195
- Polyphen-2: https://doi.org/10.1038/nmeth0410-248
- REVEL: https://doi.org/10.1016/j.ajhg.2016.08.016
- SIFT, SIFT4G: https://doi.org/10.1038/nprot.2015.123
- VEST4: https://doi.org/10.1186/1471-2164-14-S3-S3
- AlphaMissense: https://doi.org/10.1126/science.adg7492
- EVE: https://doi.org/10.1038/s41586-021-04043-8

- ESM1b: https://doi.org/10.1038/s41588-023-01465-0
- CHASMplus: https://doi.org/10.1016/j.cels.2019.05.005

Methodology

Whenever a new classification score of variants is proposed it is always convenient to evaluate how this new method compares with other scoring methods on an evaluation set that reflects the ground truth and guarantees that none of the methods has an artificially inflated performance. One would like to compare against methods intended to solve the same problem. Even the experimental saturation mutagenesis assays that assess the effects of mutations in experimental systems (cell lines or mice) do not necessarily test their oncogenic effect in the human organism. This is why we want to compare the performance of boostDM models, experimental saturation mutagenesis, and bioinformatic tools against ground truth labels built upon observed and synthetic mutations.

Here we take as ground truth driver mutations the mutations that have been observed in cancer genes across human tumors and which have a consequence type with high enrichment in driver mutations – asserted from dN/dS analysis. This set might include some non-driver mutations. On the other hand, synthetic mutations generated following the mutational profile of neutral mutagenesis across samples in the same tumor type constitute a good approximation of the set of passengers. To build an evaluation set, we used mutations in the test sets held out from training across boostDM base classifiers as drivers (observed, high dN/dS consequence type) and nondrivers (synthetic mutations generated in accordance with the background mutational profile). The evaluation set is built by taking all the mutation instances from the testing splits at each base classifier, with each mutation annotated with their mutational features, the boostDM score cast by the corresponding base classifier and the ground truth label.

We built an evaluation pool by taking all the mutations from the testing splits of each base classifier, each mutation annotated with their mutational features, the boostDM score cast by the base classifier and the driver/nondriver ground truth label. By using held-out testing mutations and base classifiers as predictors we reduce the leakage at prediction time for the sake of an unbiased evaluation.

On the one hand this evaluation strategy is expected to still bear some information leakage because we do not completely overrule the possibility that a mutation used at training is present in their corresponding testing splits, but we enforce that if this happens the mutation has unique instance in the test set, i.e. we drop duplicates in the test set. On the other hand, our evaluation strategy makes use of the suboptimal predictions cast at the base classifier level (before aggregation of the base classifiers into a consensus model) which would play against performance inflation.

We mapped all the scores to be tested onto the evaluation set, and assessed the performance of each against the ground truth labels by computing the precision recall curves, given the fact that none of the scores admitted a clear cut dichotomic interpretation as they are given as continuous numerical scores.

References

1. Pollard, K. S., Hubisz, M. J., Rosenbloom, K. R. & Siepel, A. Detection of nonneutral substitution rates on mammalian phylogenies. *Genome Res.* **20**, 110–121 (2010).
2. Kakudo, Y., Shibata, H., Otsuka, K., Kato, S. & Ishioka, C. Lack of Correlation between p53-Dependent Transcriptional Activity and the Ability to Induce Apoptosis among 179 Mutant p53s. *Cancer Res.* **65**, 2108–2114 (2005).
3. Kato, S. *et al.* Understanding the function–structure and function–mutation relationships of p53 tumor suppressor protein by high-resolution missense mutation analysis. *Proc. Natl. Acad. Sci.* **100**, 8424–8429 (2003).
4. Kawaguchi, T. *et al.* The relationship among p53 oligomer formation, structure and transcriptional activity using a comprehensive missense mutation library. *Oncogene* **24**, 6976–6981 (2005).
5. Findlay, G. M. *et al.* Accurate classification of BRCA1 variants with saturation genome editing. *Nature* **562**, 217–222 (2018).
6. Mighell, T. L., Evans-Dutson, S. & O'Roak, B. J. A Saturation Mutagenesis Approach to Understanding PTEN Lipid Phosphatase Activity and Genotype-Phenotype Relationships. *Am. J. Hum. Genet.* **102**, 943–955 (2018).
7. Martincorena, I. *et al.* Universal Patterns of Selection in Cancer and Somatic Tissues. *Cell* **171**, 1029-1041.e21 (2017).
8. Martínez-Jiménez, F. *et al.* A compendium of mutational cancer driver genes. *Nat. Rev. Cancer* **20**, 555–572 (2020).
9. McLaren, W. *et al.* The Ensembl Variant Effect Predictor. *Genome Biol.* **17**, 122 (2016).
10. Hornbeck, P. V. *et al.* PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. *Nucleic Acids Res.* **43**, D512–D520 (2015).
11. Lundberg, S. M. *et al.* From local explanations to global understanding with explainable AI

for trees. *Nat. Mach. Intell.* **2**, 56–67 (2020).

12. Giacomelli, A. O. *et al.* Mutational processes shape the landscape of TP53 mutations in human cancer. *Nat. Genet.* **50**, 1381–1387 (2018).
13. Kotler, E. *et al.* A Systematic p53 Mutation Library Links Differential Functional Impact to Cancer Mutation Pattern and Evolutionary Conservation. *Mol. Cell* **71**, 178-190.e8 (2018).
14. Ursu, O. *et al.* Massively parallel phenotyping of coding variants in cancer with Perturb-seq. *Nat. Biotechnol.* 1–10 (2022) doi:10.1038/s41587-021-01160-7.
15. Boettcher, S. *et al.* A dominant-negative effect drives selection of TP53 missense mutations in myeloid malignancies. *Science* **365**, 599–604 (2019).
16. Matreyek, K. A. *et al.* Multiplex assessment of protein variant abundance by massively parallel sequencing. *Nat. Genet.* **50**, 874–882 (2018).
17. Bandaru, P. *et al.* Deconstruction of the Ras switching cycle through saturation mutagenesis. *eLife* **6**, e27810 (2017).
18. Di Tommaso, P. *et al.* Nextflow enables reproducible computational workflows. *Nat. Biotechnol.* **35**, 316–319 (2017).
19. Chen, T. & Guestrin, C. XGBoost: A Scalable Tree Boosting System. in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 785–794 (ACM, New York, NY, USA, 2016). doi:10.1145/2939672.2939785.
20. Eilbeck, K. *et al.* The Sequence Ontology: a tool for the unification of genome annotations. *Genome Biol.* **6**, R44 (2005).
21. Arnedo-Pac, C., Mularoni, L., Muiños, F., Gonzalez-Perez, A. & Lopez-Bigas, N. OncodriveCLUSTL: a sequence-based clustering method to identify cancer drivers. *Bioinformatics* doi:10.1093/bioinformatics/btz501.
22. Tokheim, C. *et al.* Exome-Scale Discovery of Hotspot Mutation Regions in Human Cancer Using 3D Protein Structure. *Cancer Res.* **76**, 3719–3731 (2016).
23. Martínez-Jiménez, F., Muiños, F., López-Arribillaga, E., Lopez-Bigas, N. & Gonzalez-Perez,

A. Systematic analysis of alterations in the ubiquitin proteolysis system reveals its contribution to driver mutations in cancer. *Nat. Cancer* 1–14 (2019) doi:10.1038/s43018-019-0001-2.

24. Punta, M. *et al.* The Pfam protein families database. *Nucleic Acids Res.* **40**, D290–D301 (2012).
25. Lindeboom, R. G. H., Supek, F. & Lehner, B. The rules and impact of nonsense-mediated mRNA decay in human cancers. *Nat. Genet.* **48**, 1112–1118 (2016).
26. Lindeboom, R. G. H., Vermeulen, M., Lehner, B. & Supek, F. The impact of nonsense-mediated mRNA decay on genetic disease, gene editing and cancer immunotherapy. *Nat. Genet.* **51**, 1645–1651 (2019).
27. Shapley, L. S. A Value for n-Person Games. in *Contributions to the Theory of Games, Volume II* (eds. Kuhn, H. W. & Tucker, A. W.) 307–318 (Princeton University Press, 2016). doi:10.1515/9781400881970-018.
28. Satopää, V. A. *et al.* Combining multiple probability predictions using a simple logit model. *Int. J. Forecast.* **30**, 344–356 (2014).