Using machine learning to predict pathogenicity of genomic variants throughout the human genome

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)

im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin
von

M.Sc., Philipp Rentzsch

(komm.) Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Peter Frensch

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät
Prof. Dr. Elmar Kulke

Gutachter:
1. Prof. Dr. Uwe Ohler
2. Prof. Dr. Dominik Seelow
3. Prof. Dr. Peter Krawitz

Tag der mündlichen Prüfung: 26.04.2022

Abstract

It is estimated that more than 6,000 different genetic diseases are caused by genomic variants. This can happen in many possible ways: a variant somewhere in the genome may stop the translation of a protein, interfere with gene regulation, or alter splicing of the transcribed pre-mRNA into an unwanted isoform. To pinpoint the causal variants of a disease, it is necessary to investigate all of these processes and evaluate which is the most likely to result in the deleterious phenotype. A great help in this regard are variant effect scores. Implemented as machine learning classifiers, they integrate annotations from many different resources to rank genomic variants in terms of pathogenicity.

Developing such a variant effect score requires different steps: annotation of the training data, feature selection, model training, benchmarking, and finally deployment for the model's application. Here, I present a generalized workflow of the entire process, implemented as four Snakemake pipelines. The underlying framework makes it simple to configure how information is converted into model features, enabling the rapid exploration of different annotations. The workflow further implements hyperparameter optimization and model validation steps. For deployment, a selected model is applied to obtain the genome-wide score distribution and can be released as an offline service, enabling everyone to score individual sets of genomic variants.

The workflow is applied to train Combined Annotation Dependent Depletion (CADD), a popular variant effect model that is able to score SNVs and InDels genome-wide. I show that the workflow can be quickly adapted to novel annotations by porting CADD to the latest genome reference build GRCh38. Further, I demonstrate the integration of deep-neural network scores as features into a new CADD model, improving the annotation of RNA splicing events. Finally, I apply the workflow to train multiple variant effect models from training data that is based on variants selected by their allele frequency. With 70 million training instances and more than 1,000 different features, these represent the largest data sets currently used for variant pathogenicity prediction.

In conclusion, the developed workflow presents a flexible and scalable method to train genome-wide variant effect scores based on individually specified training data and annotation sets. The developed scores are freely available via a web service and as offline scoring scripts from `https://cadd.gs.washington.edu` and `https://cadd.bihealth.org`.

Zusammenfassung

Geschätzt mehr als 6.000 verschiedene Erkrankungen werden durch Veränderungen im menschlichen Genome verursacht. Vielerlei Gründe können dafür ursächlich sein: Eine Variante an einer Stelle des Genoms kann die Translation eines Proteins stoppen, die Genregulation stören oder das Spleißen der transkribierten pre-mRNA in eine unerwünschte Isoform begünstigen. All diese Prozesse müssen untersucht und bewertet werden, um letztlich die zum beschriebenen Phänotyp passende Variante zu ermitteln. Eine große Erleichterung in dieser Hinsicht sind Varianteneffektmodelle. Diese auf maschinellem Lernen basierenden Klassifikatoren integrieren Annotationen aus verschiedenen Quellen um genomische Varianten hinsichtlich ihrer Pathogenität zu bewerten.

Die Entwicklung eines Varianteneffektmodells erfordert ein ganze Reihe von Schritten: Annotation der Trainingsdaten, Auswahl der Modellfeatures, Training verschiedener Modelle, Bewertung und Selektion eines Modells und schließlich Anwendung in der Praxis. Hier präsentiere ich ein verallgemeinertes Workflow dieses Prozesses. Das zugrundeliegende Framework ermöglicht es mit wenigen Handgriffen den ganzen Prozess zu konfigurieren, Modellmerkmale und deren Prozessierung zu bearbeiten, und verschiedene Annotationen zu testen. Der Workflow umfasst außerdem Schritte zur Optimierung von Modell-Hyperparametern, Modellvalidierung und letztlichen Anwendung des ausgewähltes Modells durch genomweites Vorberechnen von Varianten-Scores.

Der Workflow wird zur Weiterentwicklung von Combined Annotation Dependent Depletion (CADD), einem Varianteneffektmodell zur genomweiten Bewertung von SNVs und InDels, verwendet. Durch Etablierung des ersten Varianteneffektmodells für das aktuelle humane Referenzgenome GRCh38 demonstriere ich die gewonnenen Möglichkeiten schnell Annotationen aufzugreifen und neue Modelle zu trainieren. Außerdem zeige ich, wie die Integration von Deep-Learning-Scores als Features in einem neuen CADD-Modell die Vorhersage von RNA-Spleißing-Events verbessert. Der Workflow wird weiterhin dazu eingesetzt um Varianteneffektmodelle mittels eines neuen Trainingsdatensatzes zu entwickeln, welcher aus nach Allelhäufigkeit segregierten Varianten besteht. Mit 70 Millionen Trainingsinstanzen und mehr als 1.000 Features basieren diese Modelle auf den derzeit größten Datensätzen, die zur Vorhersage von Varianteneffekten verwendet werden.

Zusammenfassend wird gezeigt, dass der entwickelte Workflow eine einfach zu skalierende und flexible Möglichkeit darstellt um genomweite Varianteneffektmodelle zu entwickeln. Die damit entwickelten Scores sind unter `https://cadd.gs.washington.edu` und `https://cadd.bihealth.org` als Web-Service und stand-alone Skript frei verfügbar.

Contents

Introduction

It is the ultimate nightmare of all new parents: Somehow, their newborn child does not develop as expected. There are apparent differences to healthy babies or significant developmental delays, outside of the variation that is usually observed. Doctors are running a general health check but cannot find anything: No nutrition deficit, no infection, no poison, no external cause. The pathology seems to arise due to a problem in the biology of the young human, a bug in the genetic code, a genetic disease.

While some genetic diseases such as chromosome duplications are well-characterized, relatively simple to detect, and sometimes diagnosed even before birth, rare genetic diseases are difficult to diagnose. It has been estimated that between 5% and 8% of all humans have a rare genetic disease (Baird *et al.*, 1988), which translates to about 400 to 600 million cases worldwide. In this context, a rare disease is defined as affecting less than one in 2,000 live births. The apparent contradiction between high overall incidence but low frequency of the individual diseases is caused by the high diversity of more than 6,000 described rare genetic diseases (Amberger *et al.*, 2019). Due to the different etiology of each disease, the correct diagnosis of a rare genetic disease can take many years, with many cases remaining undiagnosed (Taruscio *et al.*, 2015). In addition, effective treatments are available for less than 10% of rare disease diagnoses (Kaufmann *et al.*, 2018). As a consequence, rare genetic diseases are a major source of child death (Stevenson & Carey, 2004) and, in industrialized societies, account for more loss of life years than infectious diseases (Mazzucato *et al.*, 2014).

The search for causal variants, differences in the patient's genome that cause the genetic disease, has become easier thanks to genome sequencing. Sequencing reads the DNA and enables the detection of variants, differences to the genomes of other humans. Millions of variants can be identified in every genome, most of them without any consequence. All are screened carefully for those variants that are most likely to cause the disease, a process that we call variant effect scoring.

This dissertation proposes improvements to that last step: finding among millions of variants those that are causing disease. It outlines a workflow to train machine learning models that rank variants throughout the entire human genome. To this end, the workflow integrates previously classified variants with biological knowledge, based on which the variant effect is assessed. The workflow is then adapted to the existing variant effect score CADD, in order to build upon it and to generate novel predictions. I present how these predictions are better than previous versions and suggest further

steps to improve variant effect scoring.

The text is structured into nine chapters. General introductions provide foundations in the fields of machine learning (**1**) and genetics (**2**). Continuing on these prerequisites, chapter **3** introduces the concepts of variant effect scoring and reviews currently existing approaches to evaluate genomic variants. Chapter **4** outlines concepts for the problem addressed in this dissertation: training a genome-wide variant classifier. Chapter **5** presents a solution to this problem in the form of a workflow that implements variant scoring in four pipelines: annotation of a training set, training of a variant model, application of that model genome-wide and finally delivery of scored variants to users. The developed workflow is applied in chapter **6** to develop a new version of the variant effect score CADD for the most recent human genome reference GRCh38. The capabilities of the new workflow are then used in chapter **7** to extend CADD by including features specifically for RNA splicing effects. Finally, results of substituting CADD's training data set with population-derived variants are presented in chapter **8**. The thesis concludes with a discussion (**9**) of the obtained results and outlines open questions and possible next steps.

Chapters 5, 6 and 7 are based on the published manuscripts Rentzsch *et al.* (2019) and Rentzsch *et al.* (2021). All analyses, whether published in the manuscripts or only shown here, have been performed by myself. My coauthors Martin Kircher, Jay Shendure, Daniela Witten, Greg Cooper, and Max Schubach have contributed to the texts, as indicated in the author contributions section of both manuscripts. For this thesis, the entire text has been restructured and revised, sharing only individual sentences and wordings with the previous publications. All figures in the associated chapters overlap in terms of content with the previous manuscripts but were all specifically created for this thesis.

In my dissertation, I describe our recent progress in the field of genome-wide effect scoring with the help of machine learning models. While the developed models are not yet able to uncover every genetic cause of disease, they significantly improve prediction on currently known test data sets. In addition, the developed workflow enables the rapid development of better models that integrate more biological knowledge and may be optimized for yet unknown objectives. With that, I hope that the generated results can help us better understand how alterations in human genomes cause disease and differences in human traits.

1 Machine learning

Automation has been a driving force of modernization for a long time. Processes that once required manual labor are now fulfilled by machines. Computers have only increased the speed of this transition and simplified many tasks via digitization. Still, many jobs require situational judgment, which is much more difficult to accomplish without human supervision. For some tasks, computer algorithms can be designed to analyze the situation before executing an appropriate response. It requires that the solution to the problem can be broken down into a well-defined set of instructions. Such a solution is not always available or quickly gets overly complicated. A "magic wand" that is often proposed for such problems is artificial intelligence (AI). Specialized algorithms are expected to analyze and understand the problem in order to formulate a solution. What hides behind AI is a combination of specially adapted knowledge and computational models that are generated by machine learning.

Machine learning is a field of computer science that develops strategies to teach computers to perform certain tasks without explicitly stating all necessary steps. It includes many different algorithms that apply strategies for recognizing patterns, learning how to do and repeat a task, and finally, optimizing the output. The result of running such an algorithm is a model of the represented process that is able to produce the desired result. There is a saying attributed to the British statistician George Box that *"All models are wrong, but some are useful"* (Box, 1976), which applies very well for any machine learning model. It is implying that any model, no matter how complicated and all-embracing it may seem, will ever capture the entire nature of the process under consideration and is instead a simplification. However, some models improve our understanding of the process and enable predictions that outperform random guessing. Machine learning models are being used for many applications in automation, prediction, and discovery, such as weather forecasts, image analysis, playing games, and more and increasingly medical applications.

This chapter serves as an introduction to machine learning as is necessary for the understanding of this thesis. As such, the introduction is by no means complete and is not intended to serve as a full introduction to the entire field of machine learning and statistics. My intention is to enable the reader to follow through the application of various machine learning techniques in creating and improving models for genetic variant interpretation. For an excellent and in-depth overview of the field, the interested reader may be directed to "An introduction to statistical learning" (James *et al.*, 2013).

1.1 Supervised machine learning

The arguably most common form of machine learning is learning from examples. An algorithm is presented with a task and a set of instances for which the task has already been solved. The objective of the algorithm is then to derive rules so that the same task can be performed on further instances. These rules are generated through what is called training, the repeated, iterative application of the developing model to the training instances, followed by an error analysis that leads to model optimization so that the execution of the task is improved. This form of machine learning is referred to as supervised because the objective that should be achieved is given by the training instances and the final performance can be evaluated based on a similar set of examples by the supervisor.

Basic terminology

A supervised machine learning problem is defined by a set of examples or instances N. Each instance $i \in N$ consists of a feature vector x_i and a label y_i. x_i is a vector that contains the features J of the instance. Each feature $j \in J$ represents a property of the instance. The label y_i is the property that should be predicted. It has to be noted that while in most cases y_i is a direct and independently identifiable property of x_i, it can also be a derived value that is only meaningful within the entirety of the data set, such as a similarity to other instances and belonging to a certain group of instances.

The general objective of the algorithm is to find a function m that maps x_i to y_i for any $i \in N$ as close as possible. This means that the difference between the predicted result $\hat{y}_i = m(x_i)$ and y_i should be minimized over all instances. The difference between prediction $\hat{y}$ and label y is measured via the loss function $L(y, \hat{y})$. The machine learning algorithm minimizes the total loss of the training set $l = \sum_{i \in N} L(y_i, \hat{y}_i)$.

All $|N|$ instance vectors x_i are summarized in the training matrix X or more specifically X_{train} with $x_i \in X, i \in N$. The labels and predictions of all instances are summarized in Y and $\hat{Y}$ respectively.

Regression and Classification

Possible label values are an important property of any machine learning problem. Generally, problems can be divided into regression and classification problems. Regression describes the prediction of a continuous numerical variable that can, in principle, have

any rational value ($y \in R$), although further constraints may apply depending on the type of problem. This contrasts with classification, which decides between defined states or classes. Classification means that the label variable is categorical, either class A or class B, encoded as 0 or 1: $\hat{y}_i \in \{0, 1\}$. Most classification algorithms can be adapted from making a single yes-or-no decision, called binary classification, to distinguish many different categories in a multiclass prediction.

Predicted label values are not necessarily limited to the training label values The prediction of a binary classification model can instead be expressed as continuous class likelihood between 0 and 1. Thus, the division into regression or classification is based entirely on the input labels. It is important for the choice of algorithm and hyperparameters, as well as for the design of model performance metrics. Somewhat confusingly, the term regression is also used to describe machine learning algorithms like linear regression, and more generally describes the process of approximating a continuous variable.

Model architecture

There are various linear and nonlinear ways of how a machine learning model evaluates and integrates many different features into a single prediction, i.e. how a model makes a decision. The two most prevalent of these are linear weighting of features in comparison to each other and defining thresholds for each feature. This distinction enables different model structures. Linear weighting is used to create linear combinations of all features of a data set: Each feature j is multiplied by a single parameter a_j, all multiplications are summed up and added to an offset b:

$$\hat{y}_{i,linear} = m_{linear}(x_i) = \sum_{j \in J} (a_j x_{i,j}) + b$$

The prediction is often transformed by a nonlinear activation function. These linear models (Fig. 1A) scale relatively well to many features and, by using the absolute value of a feature, benefit from gradual features that encode multiple, escalating levels or gradients instead of a single binary decision. This is in contrast to threshold-based decision making, which evaluates features according to their relative value rather than absolute value. The advantage of these decisions is that they cannot only be added up but also be stacked, allowing for more complex "if this feature is greater than value R, then what about feature J"-structures. This structure is used in decision trees (Fig. 1B), where many decision layers are stacked. The first decision, where the evaluation starts,

is called the root. Based on evaluation at the root, any instance passes through other branches where the next decisions are made. After the final decision for the instance is made, the instance reaches a leaf node that assigns a score, the predicted output of the model for all instances that reach that particular node.

The lack of ability to learn feature interaction has always been a limitation of linear models. Therefore, there has long been active research into combining multiple linear layers by using one layer to predict meta-features that serve as input for the next layer. The principle trick has been to process the result of each layer in an activation function. so that the next layer does not directly compute the linear result of the previous layer. At the same time, the entire network is kept differentiable in order to enable a gradient descent based training process. In recent years, multi-layered networks (Fig. 1C), better known as neural networks and deep learning, have become omnipresent in machine learning. This has led to an explosion of available architectures that are transforming the applications of machine learning in many areas like image processing.

It is possible to train many different models with various structures for any machine learning problem. Each of these will cover certain parts of a problem better or worse than others. Accordingly, it can be useful to join multiple predictions. For example, multiple tree models are combined into "forests" of many trees, which are often trained iteratively so that later trained tree models complement the prediction of the first trees. More generally, ensemble models combine the predictions from two to any number of already trained models, either by averaging over all models or by training a meta-model that uses the different model outputs as features for its prediction.

Finding the optimum: Algorithms for supervised machine learning

After the model structure has been set, the model is initialized by defining the starting values of the model parameters. Typically, no prior knowledge exists so that two options are available: Either all parameters are initialized as zero and the same output is predicted for all instances, or all parameters are set to random values and therefore the model initially generates a random prediction.

From this starting point, the objective of the machine learning algorithm is to minimize the loss of the instances in the training data set. In principle, this would be possible by repeatedly selecting random parameters, checking the loss, and reverting the change if no benefit for the model is observed. Such a strategy, known as random search, is used in the field of evolutionary algorithms. While random search will eventually generate a

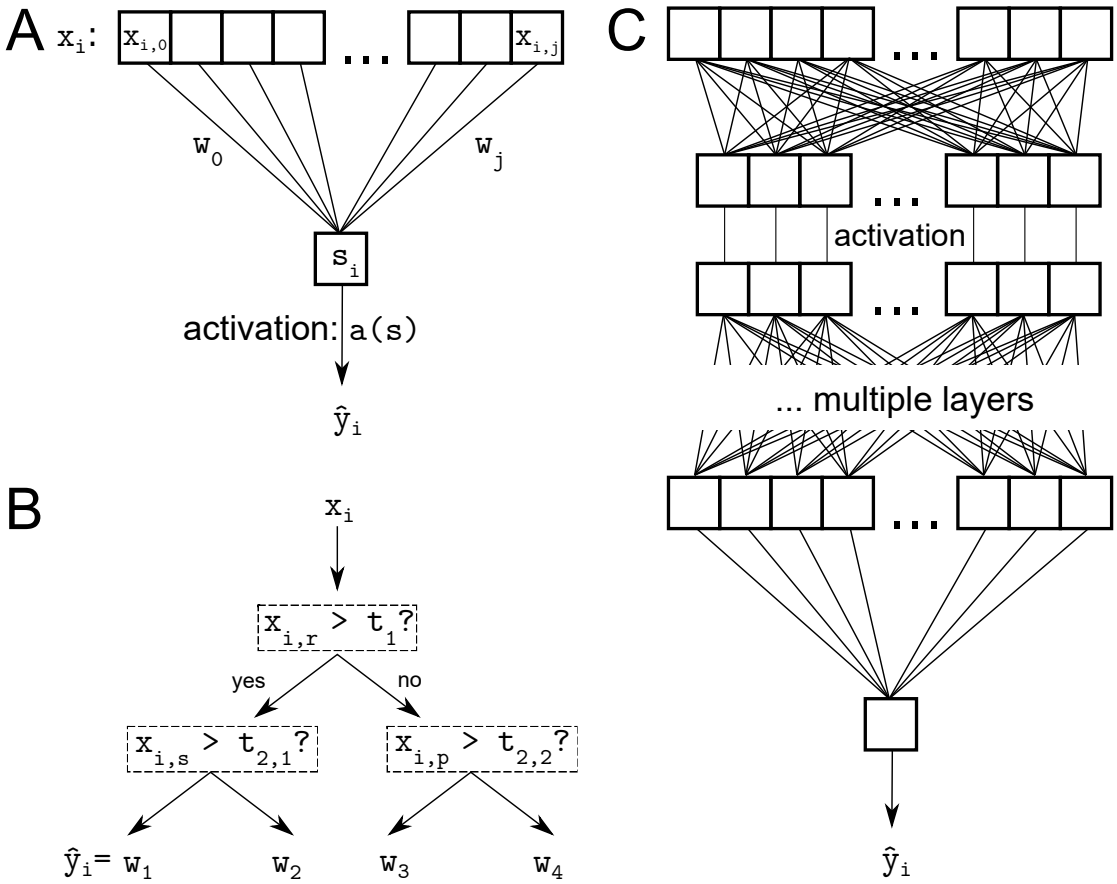

Figure 1: **Basic machine learning architectures.** A: Linear model, B: Tree Model, C: Neural network. All model architectures are read from top to bottom, as the input features are provided at the top and the model prediction is retrieved at the bottom.

good solution, it is slow and in most cases, advanced optimization algorithms converge much faster.

A requirement for gradient descent, a fast and popular optimization algorithm, is that the model is differentiable relative to its parameters. When optimizing a single parameter in order to reduce the training loss, any change applied to that parameter must be in the opposite direction of the first derivative of the loss function with respect to the parameter. In principle, it is possible to move that parameter slightly in that direction and then move on to the next parameter. A faster way is to update all parameters at once by calculating the derivative as a function of all parameters (pseudo code in Sc. 1). The magnitude of the first derivative indicates how much changing each parameter

affects the result. The gradient over all parameters puts all updates in relation to each other. By introducing a constant learning rate by which the gradient is multiplied, a factor is introduced that indicates how much the parameters should be updated. The parameters are updated with the calculated step before the new gradient is analyzed in the next iteration until the training is eventually stopped.

Sample code 1: Gradient descent pseudocode

```
# iterative function calls until training is stopped
function updateWeights(parameters, offset, X, Y, learning_rate)
{
  total_parameter_derivative = [0] * length(parameters)
  total_offset_derivative = 0
  for x_i, y_i in (X, Y)
  {
    # Calculate partial derivatives and add to total
    total_parameter_derivative += lossDerivative(parameters, x_i, y_i)
    total_offset_derivative += lossDerivative(offset, x_i, y_i)
  }
  # subtract derivatives to decrease loss
  parameters -= total_parameter_derivative / length(X) * learning_rate
  offset -= total_offset_derivative / length(X) * learning_rate
  return(parameters, offset)
}
```

While the first derivative signals how the prediction is affected by a parameter change, the second derivative can be used as an estimator of how much the update will improve the result. Calculating all second derivatives, called the Hessian matrix, is more complicated to compute though. Instead, it is possible to use an approximation of the Hessian to find better parameter values. This principle is used in the popular BFGS algorithm, named after the initials of its inventors (Broyden-Flechter-Goldfarb-Shannon). Due to its rapid convergence (Fletcher, 1987), BFGS is used in most modern gradient descent library implementations. A variation popular for large data sets is stochastic gradient descent (SGD). With SGD, the gradient is computed for only a few instances in each iteration rather than for the entire training data set. This allows a single update step to be computed much faster. SGD is popular in deep learning because computing gradients for neural networks is expensive and many small updates have shown good results. Due to the stochastic nature of the parameter update in SGD, that can lead to oscillating parameter values, it is often combined with a momentum term. Momentum is the stored average gradient from previous update steps that is gradually adapted by decay and addition of new updates. It stabilizes the parameter update and leads to better convergence. Due to the popularity of gradient descent, there are a large number of further variations that improve the optimization in various ways. Because of that, the

algorithm has been adapted to train many different types of models, including logistic regression problems, support vector machines, and deep neural networks.

The output of a generalized linear model, the linear combination of the input features, can be any rational number. In classification problems, this linear model output is mapped via an activation function into a given range like [0,1]. A prevalent example of an activation function is the logit function $\sigma(t) = \frac{e^t}{e^t+1} + \frac{1}{1+e^{-t}}$ with the linear model output t. The choice of the activation function is important, as its integral determines the loss function for gradient descent. For logit activation, this is the logistic loss function $\Sigma(t) = \frac{1}{\log(2)} \log(1 + e^{-t})$ and the reason why this type of regression analysis is called logistic regression. The choice of the activation function depends on the expected error distribution of the instance labels and regulates the algorithm's focus between minimizing the number of outliers and minimizing the average error of all instances.

A relatively simple method to extend linear models to higher orders works via the kernel trick. This principle, which forms the basis for support vector machines (SVMs), symbolically transforms the input dimensions via the kernel function into a higher-dimensional space where the training classes can ideally be separated by a single hyperplane. When using a linear kernel that does not generate additional dimensions, SVM implementations are almost identical to logistic regression and typically differ only in the use of hinge loss instead of logistic loss.

Agnostic of linear feature combinations are decision trees, a type of classification model that resembles a flowchart. Decision trees are applied by asking successive yes-or-no questions about an instance, with each question depending on the answer (or decision) to the previous question. In this way, all instances are divided into groups that share a classification. Decision trees are not specific to machine learning, as classification splits can obviously also be chosen manually by an informed agent. To distinguish decision trees trained by machine learning, these are generally referred to as classification and regression trees. Trees cannot be learned directly via gradient descent because each decision split has to be made through a choice that is not directly differentiable. Instead, each split is determined algorithmically by computing information criteria like entropy or Gini coefficient for each feature and selecting the feature that separates instances best. A tree is created from top to bottom by selecting the root decision first and then proceeding to the two possible decision branches. On each branch, a new decision split is then created that divides the training instances that have proceeded to that point. Additional decision branches are added until a certain depth threshold is reached or

the information gain from a new decision does not reach a specified threshold. The information content, or a similar metric, of all instances that reach a terminal node is used for the output of the tree model. As mentioned earlier, decision trees are rarely trained individually and are mostly used in ensemble models of many different trees in a forest. Training multiple, singular trees will result in the same model each time as the training algorithm does not include random parameters. To obtain multiple different trees that complement each other, randomness can be induced by training multiple trees in parallel, randomly withholding training instances from each tree, or limiting the number of features available for each split. This creates a so-called random forest of trees that each lead to slightly different predictions. The predictions from all trees are then aggregated into a single prediction.

Unless specified otherwise, each instance in a training set is equally important and weighted to the same amount in every training iteration. Equal weighting may not be optimal for problems where some instances are relatively simple to predict while others require more attention to be correctly classified. In that case, harder-to-predict instances can be emphasized in training via boosting. The original idea behind boosting was to combine multiple weak learners that do not yield good model performance on their own into an ensemble model with better model performance (Kearns & Valiant, 1989). Unlike random forests, a boosting model is not learned as the sum of multiple weak learners but is trained iteratively, with each iteration starting from the predictions of the previous iteration. In each training iteration of an adaptive boosting classifier (AdaBoost, Freund & Schapire, 1995), the predictions of all instances are evaluated. Instances that were classified correctly are decreased in weight while the weights of misclassified instances are increased. The new weights are used in the following iteration to focus the next added learner on those subsections of the machine learning problem that have not been considered in the present model. Gradient boosting (Friedman, 2001) is a similar boosting algorithm. Unlike AdaBoost, it does not change the weights of the individual instances but rather computes the residual loss of each instance. The model update is then performed by gradient descent on this instance loss. The individual weak learners in both AdaBoost and gradient boosting are usually implemented using decision trees.

Creating a training data set

The success of any attempt to solve a problem with machine learning depends largely on quantity and quality of the training data set. In general, the more data there is, the more examples there are to learn from. The training data set should be representative of the problem to be solved, with training instances coming from the same distribution as the final application of the model. To give an example: When classifying dog and cat images, it is problematic if dogs and cats are always in the center of the training images when the same cannot be expected in the application. It is also important to consider limits for predictions on extreme examples, such as applying a prediction model of family house prices to a castle.

Augmentation is a popular technique for increasing the size of a training set. It means that training samples can be included multiple times as quasi-independent training instances via simple transformations or the addition of noise. An example of this is flipping or rotating images in a dog-cat classifier. Possible transformations for augmentation depend on the posed problem and data formats. Generally, augmented instances should not simply cluster with their originals.

Even the best human-annotated training set will contain misclassified instances. While inevitable, misclassification is not a huge problem as long as it is not biased towards one output value or class. It can be helpful to know the magnitude of this human-level error rate, as it represents an upper boundary on algorithmic prediction performance on this type of data. In other words, a model with 100% prediction accuracy will reproduce the human-level error rate, while a perfect model, better than the human classification, cannot reach perfect performance if human mistakes in the training data count toward its error rate. Comparing model performances within the range of the human-level error rate is therefore limited.

1.2 Feature engineering

The process of finding, analyzing, and adapting sources of information that describe the data in a machine learning project is called feature engineering. It starts with a thought process on what type of information may describe the problem best and may be useful for the model. Once these have been identified, the analyst searches annotations, streams or sources of data that provide that information. These annotations have to be screened for their benefits, including availability for all instances, inherent biases,

accuracy, used units, and overlap with already existing annotations. The data structure of selected annotations has to be converted to a format that is needed for the computer algorithm. After integration into the machine learning model, an iterative process of checking how well the feature is improving the model, how it may be refined, and which other features may be useful to maximize performance follows.

Feature engineering, similar to training data set preparation, is hard to generalize. The main reason for this is the diversity of input types that need to be interpreted before they can be integrated into a model. For example, color may be a categorical feature in a data set of cars, but a numerical feature when predicting the outcome of a physical experiment. Or in one data set, one brand of cars has defined colors by an international standard while another manufacturer uses their own color naming scheme. The generation of good features is therefore very often a complex and iterative step and can occupy a significant fraction of the total development time. It does hence not come as a surprise that features are frequently adapted from previous projects. Similarly, we have also found that in project descriptions features are sometimes kept intentionally vague when they provide a main advance in the improvement of a novel model.

Variable types

Every feature is representing a particular property, that describes an instance and is later used for predicting the target outcome. The value of the feature is obtained from an information source, the annotation. Each property is unique and describes the instances in a certain way, like length in centimeters, color name, or position in a relative context. Accordingly, there are many different types and formats of features like strings, numeric measurements with units, lists, and graphs. However, these human language terms often cannot be used directly by a computer. Instead, the value for each feature of an instance is encoded in one or multiple numeric variables. The variables of all features of an instance are stored in one vector, x_i. This vector always has the same dimensions and internal order, meaning that element n of the vector x_i always represents the feature n of the instance i. In terms of encoding, there are three main types of features: binary, categorical and numeric, as well as several subtypes, distinguished by attributes and possible values (Tab. 1). While the decision for a type of encoding is usually simple, it is important for how the feature is interpreted in the model.

The simplest of these types is binary when the instance either has a feature or not. It is encoded, 0 or 1, in a single variable. An example of this would be if a house has a

Table 1: Encoding different features types

Feature Type	Example	Encoding	Possible values
Binary	house has chimney: yes	scalar	0 or 1
Categorical - Nominal	house color: yellow	vector	0 or 1
Categorical - Ordinal	house low energy rating: 75%	scalar	each level represents a numeric value
Numeric - discrete	number of windows: 8	scalar	any integer
Numeric - continuous	room height: 2.8 meters	scalar	any decimal number

chimney or not. It is further used for features where only two valid options are possible, like the active light of a two-color traffic light.

If there are more than two states predefined states that the property can have, the feature is of a categorical type. Examples are the color shown on a three-color traffic light or the nucleotide in a DNA sequence. Categorical features are further distinguished into nominal categorical, with categories that do not have an internal order like different species or names, and ordinal categorical where such an order exists, like ranking on a list or states of severity of a disease. This distinction is important for the encoding of the feature in one or multiple variables in the feature vector x_i. Nominal-categorical features are usually encoded, similar to a multiclass label, in several binary "dummy variables" with each binary variable encoding if one particular category is true or not. While it would technically be possible to store information about a finite set of possible categories in fewer bits, this enables direct access to the single category via a single coefficient without decoding the different categories first. By definition, categorical features can only have one value, so that exactly one of those binary variables is true, an encoding that is called one-hot-encoding. For some features, multiple categories can be selected in the same instance. In that case, multiple or also none of the binary variables can be true. While such a feature is very similar to a categorical one, in the implementation this is better described as a series of binary features.

In an ordinal categorical feature, in contrast to nominal categories, the different categories are assigned to relative ranks, with each rank specified as a distinct numeric value. While the same encoding as nominal categorical can be chosen for such categories, the order enables that the numeric values are encoded in a single variable, which can have defined values equivalent to the different ranks in an order. This has the advantage that the feature does not generate as many variables as there are categories of which most have zero value anyways but has the disadvantage that the different numerical

weights have to be chosen explicitly.

Very similar to the encoding of an ordinal categorical feature is the discrete numerical feature. Discrete numerical means that the value of the feature is usually of an integer value, with a limited number of possible states between two amounts. The difference between the ordinal categorical and discrete numeric is that the latter is a number by definition and the set of possible values is not necessarily a finite set, while the numbers of ordinal categories are specifically set and limited to a finite set of possible states. To exemplify, the number of kids in a family is a discrete numeric feature while the rank of relative age, like being the oldest child in a family, is an ordinal categorical feature. Numeric features do not have to be discrete but can be fractions. In the case of a continuous numeric variable, features are not limited to integers and are encoded in floating-point variables. These general types are however not always strict and blend into each other, like continuous one-hot-encoding where a total sum of 1 is distributed over multiple categorical variables.

While encoding of the variables is important for the handling of the values, it is important to keep in mind that in software development practice, the concatenation of all features in a single vector means that all variables are converted into a single variable type. Common, because able to store every variable type from binary to continuous numeric without considerable information loss, is the use of 16 to 64 bit floating point variables. Most variables in such a vector are nonetheless quasi-binary.

Transformations

The default distribution of a numeric feature is not always ideal for the solution of the problem. This is mostly the case for features that are not linearly correlated to the outcome of the label but in some other relationship like quadratic or exponential. Transforming a feature to a distribution that maximizes linear correlation to the label outcome can improve the uptake of a feature in a machine learning model tremendously. Some transformations, like taking the absolute value of a feature or capping a score at maximum or minimum thresholds are not reversible. This means that the transformation can result in a loss of information, as more than one input value maps to the same output value. As such, the transformation does not necessarily result in a change of the feature distribution but serves as a definition of the features range. Transformations overlap with the field of scaling, as described in the section about regularization, and generally serve a similar purpose.

Encoding

Encoding many categorical features with numerous categories can cause instance vectors with only a small number of variables different from zero. For computational reasons, all variables in the instance vector are usually stored in the same data type. Floating-point variables are preferred over single-bit binary variables, as the latter would result in a loss of information for many features. The amount of computational storage space needed to save the entire data set in encoded format is therefore inflated. A solution for this problem are sparse matrix formats. In comparison to the default "dense" $N \times M$ matrix where each row-column coordinate is stored in the data array, only those variables that are different from the default value, usually zero, are stored in the matrix object. Retrieving a value of such a matrix amounts to a lookup of whether the variable is stored in the matrix and if it is not found returning the default value. There are different types of sparse data format, each with different advantages and disadvantages. Examples are the dictionary of keys (DOK) that performs lookups in a hash table, and compressed sparse rows (CSR) that stores data via two consecutive pointers. Converting from a dense to a sparse matrix format and back again requires some computational time. This initial delay can be compensated, depending on the data sparsity, by faster saving and loading times of the entire matrix to a disk, as writing and reading of data is a bigger time constraint than conversion. Some machine learning implementations also support sparse matrix formats, which enables faster and especially more memory efficient computation than with normal dense matrices.

Imputation

What happens if an entire feature is not available for an instance or has an unknown value? If possible, because relevant only for a small fraction of all cases in training, one solution is always to ignore the instance. If that is not possible, for example because we suspect that the absence of the feature is non-random, the feature has to be imputed with a suitable value. One way to infer a value is by analyzing similar instances via a nearest neighbor search. Depending on the type of problem, this can be a good solution. However, very sparse features may lead to wrong labels being attached to many instances and thus associated mistakenly with a prediction. If we cannot impute the feature value like this, we define a default value for our feature that gets imputed to every case with an invalid value. This value can be the mode, mean or median of that feature of all other instances with a defined value. Or it is some value that is expected or adequate,

such as if a feature is missing in an instance because that particular instance cannot have that feature. For example in a data set of houses, the feature "number of floors" could be set to 1 if undefined or in a data set of vehicles, the feature "engine power" could by default be set to 0 if the data set also includes bikes. As these examples show, how a feature's imputation is encoded is dependent on the feature, machine learning algorithm, and specific problem and often takes a lot of time to evaluate.

Feature engineering for different machine learning algorithms

The level of detail and structure in which features are best engineered depends on the machine learning algorithm applied. For example, decision trees built on yes-or-no decisions do not change by relative scaling of a numeric feature. Instead, the question evaluated by the algorithm is always if the value of a variable is bigger than a given threshold. In contrast, scaling is very important for linear models where an extreme outlier of a single feature may change the predicted outcome of the entire model. Decision trees are also able to detect interactions between features as multiple yes or no decisions are stacked sequentially. This stacking enables a potential for structures of if feature A is yes and feature B is no, then feature C is important sequences. For linear models, such feature interactions have to be created via feature crossings. A feature crossing combines all possible combinations of the categories in the individual features and is equivalent to the multiplication of the encodings of two or more feature vectors. Since feature crossing creates large feature vectors that add at least the size of the original features to the model, this is a major source for training matrix sparsity and model complexity. Sparse features can be problematic for decision trees, as every single category has to be evaluated against the rest of the model. A possible solution are greedy search algorithms that summarize multiple feature categories into a single variable (Ke *et al.*, 2017).

One of the reasons why deep learning models hyped over the last few years is that they require less feature engineering (Goodfellow *et al.*, 2016). This is due to the hidden layers in the neural networks that transform the different input features in relation to each other. One type of neural network, the auto-encoder, is specially designed to store complex input into a low dimensional latent space of meta-variables. The latent space describes an instance in relatively few complex terms and can be decoded back into the full instance features. Unfortunately, this advantage comes at the cost that it is often hard to come up with reasoning why a neural network decides one way versus

the other. Multiple studies have shown that they can apply attacks to such a model by manipulating the input. For example, this could affect the prediction of an image classification algorithm while a human looking at the same image would not note any difference (Su *et al.*, 2019). While many different methods for feature attribution have been described (Ancona *et al.*, 2017), such problems exist for almost all machine learning applications and could develop into a big problem in the field of artificial intelligence and other forms of algorithmic decision making. While certainly not sufficient to fully mitigate such risks, reproducibility and in-depth documentation of feature engineering is important to audit, debug, understand, correct, and prevent such problems in the future.

Feature selection

While technically not a part of feature engineering, feature selection widely overlaps with many of its steps. Once a number of meaningful features are generated and a model with reasonable performance trained, it raises the question if all used features are strictly necessary for the model's best performance. This is especially important for related features that are strongly correlated. It could be argued that multiple similar features reduce the impact of the statistical error of the single feature on the final model. Unfortunately, due to the infinite number of possible solutions of an overdetermined set of equations, finding the optimal solution in a high dimensional space with a finite number of instances is prone to result in extreme model coefficients. This problem is generally known as curse of dimensionality.

To find the optimal combination of features, it would be necessary to test all possible combinations. The number of combinations rises exponentially, making it practically impossible for sufficiently large numbers. Several heuristic methods have therefore been developed to select features. As a simple strategy, random search approaches from the field of genetic algorithms can achieve significant results by iteratively experimenting with numerous combinations. In order to not only evaluate by performance but also simplicity, this is combined with methods that measure the mutual information of multiple features and from this deduct the information content of the single feature. Finally, features are rarely ignored entirely from the model, as it is often sufficient to merge multiple related features via an unsupervised method in a meta-feature. The amount of feature selection necessary depends on the applied machine learning algorithm as some, such as decision-trees, apply their own feature selection.

1.3 Selecting a good model: Accuracy, Optimization and Regularization

Once a machine learning pipeline is established, including retrieving the training data, annotating and encoding the instances, and training the model, the entire process can be run many times, limited only by the available computational resources. While this should, given full reproducibility, always return the same result, the configuration may be altered in several ways to find improvements like which features to include in the model, what type of algorithm to use, or how to set certain model parameters. This process of model selection is bound by model performance and complexity.

Benchmarking a model

In order to measure how well a model performs, it has to predict a number of labeled instances. The instances used in training the model have already been seen by the algorithm and may have been optimized for overproportionately. Therefore, any data set the performance is measured on should not be overlapping with the training set. To restrict occurring biases like common selection background or augmentation to a minimum, the theoretically best case for performance benchmark is total independence from the training set. In practice more common is a random split of the initial set of instances.

The selection of a single test set is statistically problematic due to multiple testing. Training large numbers of machine learning models with very similar performance can and will select for models that perform slightly better on the test set by chance. Even with a substantially sized test set and regularization, there is the risk that test results for the finally selected model are better than observed on an entirely novel data set. An independent performance estimate of a model is therefore only given for a test set that is hold-out during model optimization and first evaluated only after the final model has been selected. In summary, the model training process should include three stages that use separate data: model training, model validation, and the final performance test. If no independent test sets are available, the input data set is accordingly split into three none overlapping data sets. The training set is used for building and training the model. During the validation stage, various model versions that have been trained with different hyperparameter and feature combinations are evaluated and compared. The final test set is only evaluated when the entire training process is finished, and enables

an unbiased comparison of the final model to other approaches. Depending on the given problem, the size of the data set, and the expected performance, the number of instances in each of the three sets varies. Under the premise that the training set really should cover the entire problem as far and complete as possible, the training set is generally the largest of the three sets. Validation and test sets can be smaller, as they serve to get a statistical overview of how good the model is performing. It is to note that the definition of validation and test set can be confusing and is often mixed up (Ripley, 2009).

Except for memory or run time optimization, it would always be beneficial to have an even bigger input data set. The selection of the validation and test set excludes the included instances from the training process. For small training data set where each instance counts for training a good model, this can become a problem. On the one hand, the training set should be a big as possible while on the other hand validation and test set should also be big to provide profound test results. Various techniques are designed to mitigate this problem. Two of those, bootstrapping and cross-validation virtually increase the size of the test set in order to achieve smaller and better accounted for validation errors. In cross-validation, the initial training data set after split from the final test set is fractionated into N sets with equal numbers of instances per set. For each of these sets $n \in N$, a model m_n is trained by using the set n as validation set and all other $N - 1$ sets joined as training set. This way the entire training set can be used for performance validation and the single validation set can be smaller. Cross-validation is computationally more expensive though since multiple models have to be trained. Bootstrapping is an up-sampling technique, in which instances are randomly drawn from the training set with each instance being able to be selected multiple times. Under the assumption that the generated sample represents the total distribution of all instances, this can be used to create multiple test sets of instances. While these test sets are not independent, running a model test on all these instance sets gives an estimate of the uncertainty in the initial test result compared to a test on all instances.

Overfitting and Regularization

When derived from one mutual input data set, training, validation, and test set come from the same background and therefore should have the same feature and label distributions. Accordingly, any predictive task should generalize and perform equally well on all three data sets. Unfortunately, this is often not the case as performance on the training set becomes overly optimistic with increasing model complexity. The involved

process is called overfitting, as the model predictions are fitted too much to the training instances and perform worse when predicting unknown test labels (Fig. 2).

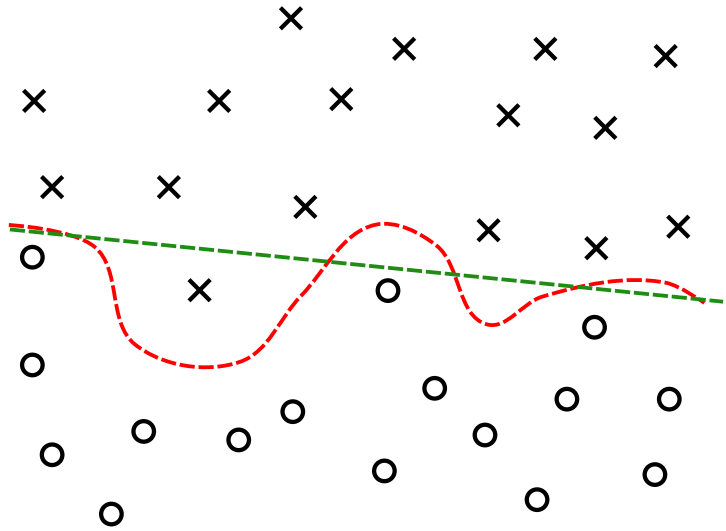

Figure 2: **Overfitting with a complex solution.** While a complex solution (red) is often better at distinguishing between instances of different classes, it may overfit to the training instances and does not generalize. This may lead to a worse performance on test instances not included in the training set. While subjectively worse on a first look, a simple solution (green) can in that case be the preferred solution.

Overfitting happens in all types of machine learning to some degree, as training instances can never represent the complete diversity of the possible input set. Nevertheless, there are several methods to regularize a model by applying constraints that monitor and limit overfitting. One approach is that while iterative training will descent with each step to an optimum in training space, its prerogative is to catch the moment with the highest accuracy in test space. Once the gain in prediction is only for the training set and leveling out for testing, the model overfits on the training data. This is often associated with a drop in test performance. The remaining gain in training performance observed is due to closely modeling the input set. Early stopping (Fig. 3) means model adaption on training and test set is monitored and the training process is stopped when the performance diverges between the two sets. Obviously, model training does not have to be stopped during run time. Rather, the model state is saved after each iteration and the stop is performed in retrospect.

Overfitting does not only happen due to the model optimization going on for too long but also when the model has more parameters than strictly necessary. In that,

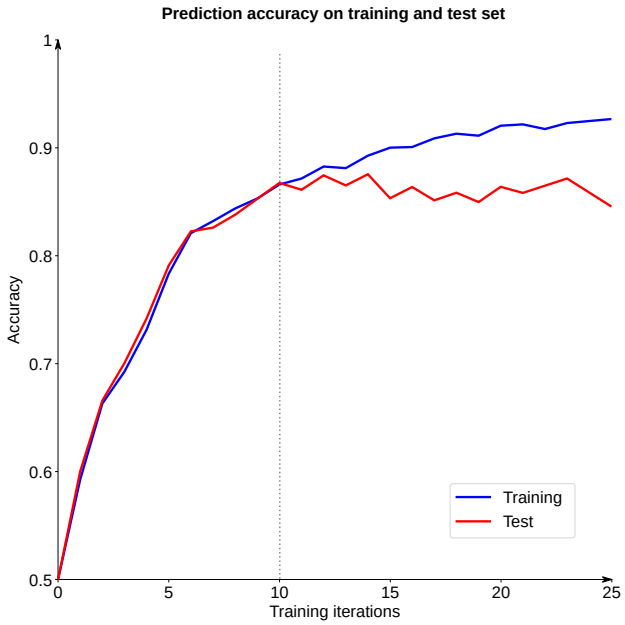

Figure 3: **Early Stopping.** For the first ten iteration, the prediction accuracy of training set (blue) and testing set (red) is highly correlated. Past that, the training accuracy keeps increasing with every iterations while the test accuracy levels out or even decreases slightly. The difference between test and training is due to overfitting the model to the training set. A good place to stop early in this case would therefore be after approximately ten training iterations.

the problem resembles an overdetermined system with many possible solutions. While overdetermination is not bad on its own, this often leads to an explosion in parameter values as the model descents to the optimal solution. One possible solution that is popular in neural networks is drop-out wherein every training interaction, a certain number of parameters is masked so that the model cannot rely on complex coadaptation of parameters. Instead, drop-out forces the model to rely on each feature for separate evidence.

The application of constraints via an added regularization term to the loss function in order to enforce smaller parameter values is the most prevalent type of regularization. With constraint, the optimization function o is formulated as the sum of the loss function L of the prediction $\hat{y}$ versus the true label y and the regularization term r of the parameter weights w, multiplied with a regularization coefficient λ that determines the

regularization strength:

$$o(\hat{y}, y) = L(\hat{y}, y) + \lambda r(w)$$

These optimization terms penalize larger parameters stronger, pushing the optimization to prefer smaller absolute parameter values. Generally, there are L_1 and L_2 regularization with intermediates like elastic-net regularization that combine L_1 with L_2. L_1 or lasso regression is determined by the L_1 norm $\|w\|_1 = |w|$. This means that regularization strength is independent of the size of the parameter and all parameters are under the same force to be as small as possible. With that, L_1 regression enforces sparse models. Only the most predictive features get assigned parameters that are different from zero and can get very large coefficients, as L_1 regularization does not penalize large parameters. In contrast to that does L_2, or ridge regression, use the L_2 norm $\|w\|_2 = \sqrt{_{i \in N} w_i^2}/|N|$. This penalization of the square of the parameter values puts more weight on large parameter values. L_2 regularization thus leads to less sparse solutions that include weight more features differently from zero. It is often preferred for complex predictions where multiple effects with very different strengths are combined into one model. The regularization coefficient determines the strength of the applied regularization, with a value of 0 equaling to no regularization. The best value of the regularization coefficient depends on the data set and generally the involved machine learning problem and is optimized over multiple orders of magnitude via hyperparameter optimization.

Both L_1 and L_2 regularization rely on the assumption that features are approximately identically distributed. This is important because increasing the value of a variable by a factor s will decrease the respective coefficient by a factor $1/s$, reducing regression weight by the same factor with L_1 and $1/s^2$ with L_2 regularization. As all features are initially considered of equal importance and thus equal regularization is applied, they should accordingly be in the same range. This is achieved by rescaling all values. Common scaling approaches are min-max-scaling and normalization. Min-max-scaling means that minimum and maximum are determined over all instances, scaled to the extremes of a defined range with all other values scaled linearly in between. In the more frequent normalization, all values are divided by the standard deviation (average squared distance from the mean) of all instances, so that the scaled output has a standard deviation of 1. Additionally, it is preferred that the mean feature value is around zero. Otherwise, the mean is a constant that is always multiplied with the feature coefficient of the linear model and only the difference from mean distinguishes different instances.

It is compensated by the model offset parameter b. The combination of normalization and mean centration is called standardization.

Generally, a data scaler is initialized as a set of transformations on all features once initially before training from the entire training data set. It is then applied to every instance that is processed through the model. A special technique is batch normalization, which is especially popular in the field of deep learning. Batch normalization means that the normalization of a feature is not established on the entire training data set but only on the batch, a small subset of the entire training data set that is loaded for one training iteration in stochastic gradient descent. The main arguments behind batch normalization are that it enables a faster (Ioffe & Szegedy, 2015) and generally more stringent training process, although exact details are unknown (Kohler *et al.*, 2018; Yang *et al.*, 2019).

Performance metrics

In order to quantify the performance, called goodness of fit, of a trained machine learning model, it is run on the labeled instances of the test set. Generally, the predicted labels are then compared to the actual labels via multiple metrics. In cases where data or labels are limited to the training data set, model prediction may be validated on surrogate labels, metrics that are correlated to the labels of the training data set and not used as a feature. The goodness of fit is always measured with regard to the test set which makes results comparable only to models within the same setting. Depending on the task and the question asked, there are several different quantification metrics.

Popular measures for goodness of fit in a regression analysis are correlation coefficients and coefficient of determination. These are metrics for how close the predicted values match to the real values. Correlation is a measure of dependence between values. As prediction is supposed to be as close to the real value as possible, the aim is to maximize correlation. Without assumption about the distribution of the real values, the most prevalent form of linear correlation analysis is Pearson correlation. Based on the mean $\mu_Y = \frac{1}{N}\sum_i^N Y_i$, covariance $cov(\hat{Y}, Y) = \frac{1}{N}\sum_i^N (\hat{Y}_i - \mu_{\hat{Y}})(Y_i - \mu_Y)$ and the standard deviation $\sigma_Y = \sqrt{\frac{1}{N-1}\sum_i^N (Y_i - \mu_Y)^2}$, Pearson's correlation is computed for prediction $\hat{Y}$ and real value Y as $\rho_{\texttt{Pearson}}(\hat{Y}, Y) = \frac{cov(\hat{Y},Y)}{\sigma_{\hat{Y}}\sigma_Y}$. The range of the Pearson correlation coefficient is from -1 to +1 with a negative correlation coefficient indicating an inverse association between prediction and actual values. On data set with few extreme and many relatively close instances, Pearson correlation can be impacted by the extreme instances due to their larger potential error. In order to get a more robust estimator,

it can make sense to compute the relative ranks of predicted and real values and then compute the Pearson correlation of those ranks. This is called a Spearman correlation. Finally, coefficient of determination is an often used metric to describe goodness of fit. It directly reflects the proportion of the total variance in the real values that is predicted by the model. Mathematically, the coefficient of determination or R^2 is the square of Pearson's correlation coefficient. Since the output or prediction of a model is always a floating-point number, regression analysis can also be applied to classification problems. This can be used to compare the prediction of several classification models to an external variable that is associated with the initial class label.

In classification analysis, there is less focus on the exact value of the label prediction. More important is that the predicted class matches the expected class with a low error rate. In a binary setting for a given threshold $t \in \mathbb{R}$, all instances with $y > t$ get predicted A and all instances with $y < t$ get predicted B. The threshold t is defined as a number somewhere between the labels for classes A and B. Typically would be $t = 0.5$ in a $y \in 0, 1$ labeling.

A confusion matrix compares the numbers of predicted and observed labels (Tab. 2&3). The overall error probability is called accuracy. Its value is the likelihood that a random instance is classified correctly by the model and is defined through the number of correctly classified instances divided by the number of all instances. Depending on the task performed, it is also important to consider error rates for every single class in isolation. To do this, one class is defined as positive P and one as negative N. An instance identified correctly is called true, resulting in true positives TP and true negatives TN. Falsely identified instances are predicted to be in the wrong (false) class. Therefore, false positives FP are predicted to be in the positive class but are negative, while false negatives FN are actually of class positive and falsely predicted as negative.

Table 2: Example classifier for two classes red and blue

	actual red	actual blue	total
predicted red	13	3	16
predicted blue	7	14	21
total	20	17	37

Table 3: Confusion matrix: relationship of actual and predicted labels

	actual Positives	actual Negatives
predicted Positives	True Positives	False Positives
predicted Negatives	False Negatives	True Negatives

The classification error rate of the positive class, $TPR = TP/(TP + FN) = TP/P$, is called sensitivity, recall or simply true positive rate. Its equivalent for the negative class is specificity, selectivity or true negative rate $TNR = TN/(TN + FP) = TN/P$. Complement to TPR and TNR respectively, there are false negative rate $FNR = FN/(TP + FN)$ and false positive rate $FPR = FP/(TN + FP)$. The sum of TPR and FNR as well as TNR and FPR is always 1. The positive predictive value or precision $PPV = TP/(TP + FP)$ is the likelihood that a positive prediction is correct. It should be noted that the assignment of positive and negative is dependent on an external definition and true positive and true negative rates can be reversed if the model task is formulated differently. Accuracy of all instances is defined as: $Accuracy = (TP + TN)/(TP + TN + FP + FN)$.

It is important to keep in mind that all these rates are mutually dependent on the classification threshold t. Increasing t decreases the likelihood that an instance is predicted to be positive, thus decreasing the number of TP and FP and increasing the number of TN and FN, which decreases TPR. The overall performance of a model is therefore assessed over all possible values of t. From the single class error rates, derived metrics describe the relationship of these error rates in comparison to each other. This means that classification metrics are computed for the same values of t, i.e. when a model has a TPR of 0.8, what is the FPR of the same model on that data set. This relationship of TPR and FPR is called Receiver operating characteristic (ROC, Fig. 4), with the graphical plot called ROC curve. The area under the ROC curve (auROC) ranges between 0.0 and 1.0, with higher values linked to better performance. Since both TPR and FPR are independent of the relative numbers of positive and negative instances, auROC is especially used in scenarios where the ratio of positive and negative instances in the final application is unknown.

Another often considered relationship is that of PPV and TPR, named after two of their synonyms precision-recall curve (PRC, Fig. 5). ROC and PRC are in most cases correlated. The area under the PRC (auPRC) is also in the range from 0 to 1 with higher values linked to better performance. If a model ROC-curve has higher TPR than another model PRC for all FPR, it will have both a higher auROC and a higher auPRC than that model. auPRC is especially useful for rating how well a small number of instances is detected in a large imbalanced data set.

Any classification measure compares to a random baseline, the performance of a null model with predictions assigning instances randomly. For most metrics, the random

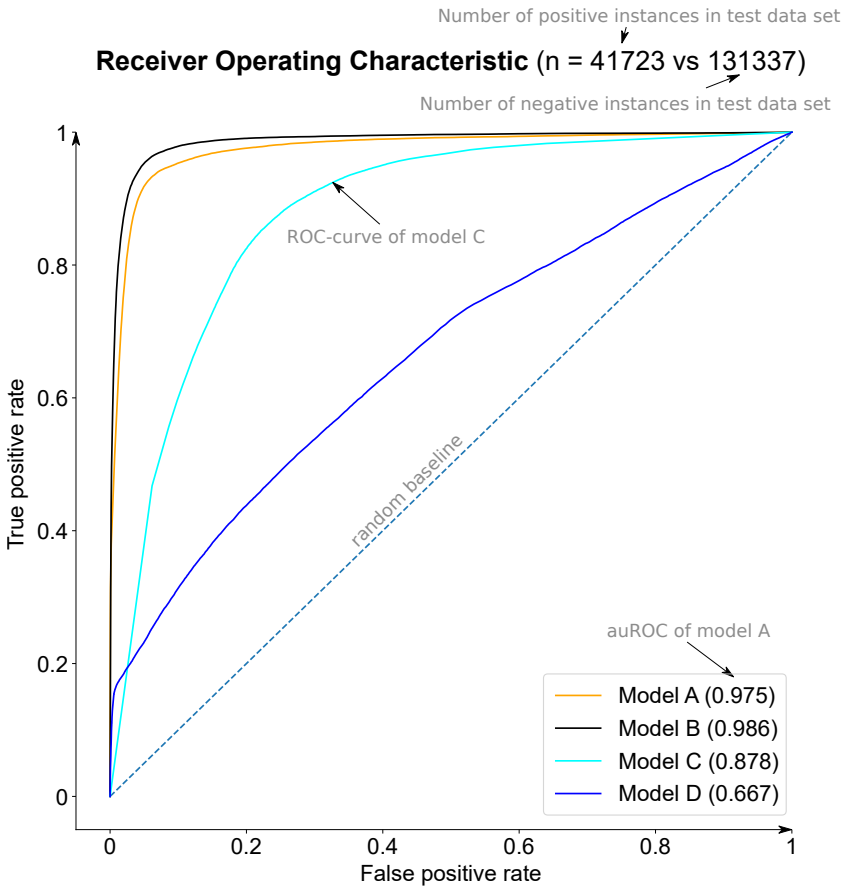

Figure 4: **Example receiver operator characteristic.** The receiver operator characteristic (ROC) displays the relationship of true positive rate (TPR) and false positive rate (FPR) for any model in a ROC curve. The area under the ROC curve (auROC) is used as a metric to describe overall model performance. A larger auROC does not mean that TPR is always better at equal FPR as shown by Model D surpassing Model C for small FPR.

baseline is defined by the class probabilities of the labeled instances. For auPRC, it is the fraction of positives in the entire test data set $P/(P+N)$. Since both TPR and FPR are only dependent on positive and negative instances respectively, in random guessing on a large sample TPR approaches FPR, resulting in a linear term with a gradient of 1 and offset 0, for all values of FPR from 0 to 1. Thus the area under the random baseline of auROC is always 0.5. Classifiers performing worse than random guessing are usually due to interpretation errors, e.g. when the instance labels have been mixed up or the prediction is negatively correlated with the input labels (Fig. 6).

In a multiclass prediction, these metrics have to be adapted. While accuracy can be calculated for all instances, TPR and the other metrics are calculated for each class

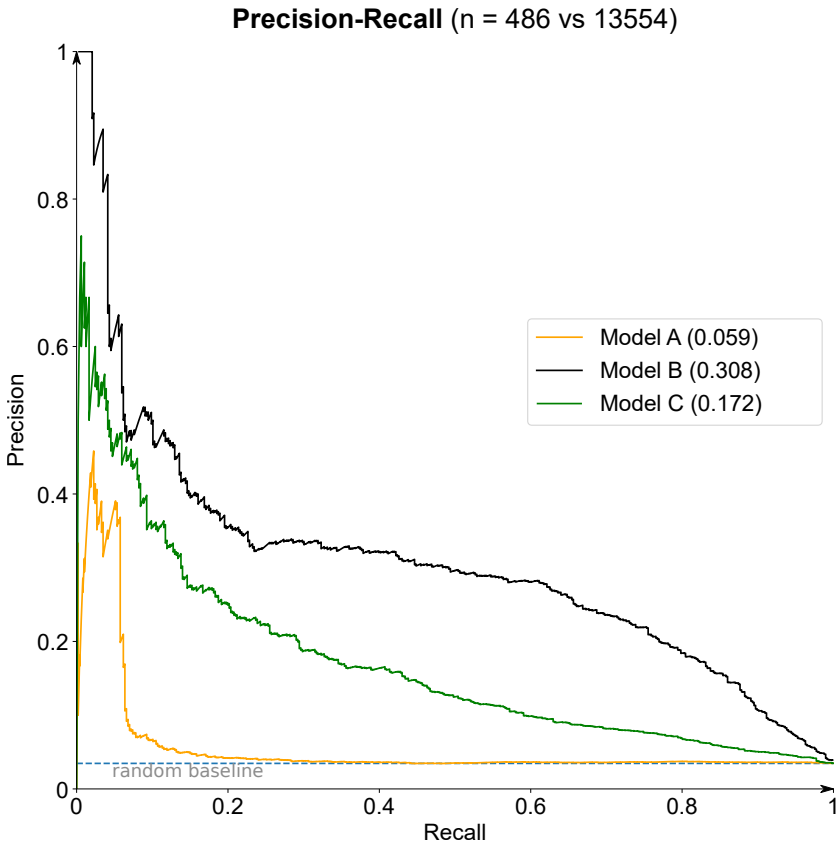

Figure 5: **Example of a precision-recall curve.** The area under the precision-recall curve (auPRC) is a metric for how well a model is at prioritizing a set of instances from a large set.

separately.

Hyperparameter optimization

When adopting a machine learning model to a novel problem, there are many hyperparameters like the strength of regression, the number of training iterations, or the number of layers in a deep neural network that have to be optimized. Based on previous projects, domain knowledge, and the available computational resources, it is usually possible to determine ranges for possible hyperparameter values. As hyperparameters are not mutually independent and certain combinations may lead to a good model, it is necessary to try out several of these combinations and then select the best combination. There are different strategies to find such combinations. The most obvious is to try out all possible combinations in a grid search. However, the number of possible combinations grows exponentially with the number of parameters. This is especially a problem in

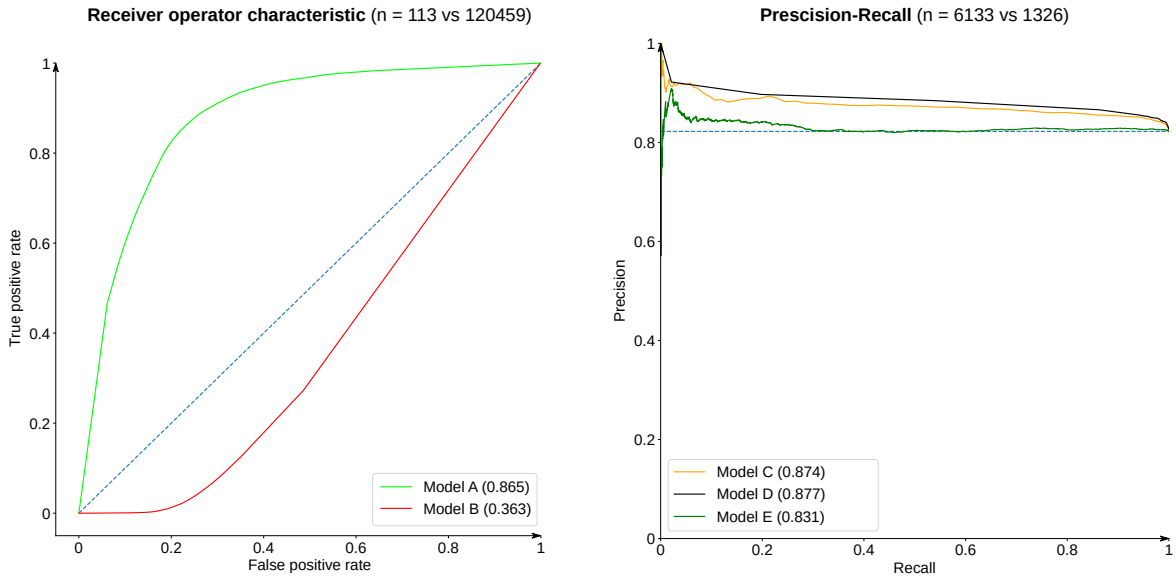

Figure 6: **Trouble with performance metrics.** On imbalanced test data set, the choice of the wrong performance metric can lead to irritating results. While this does not affect ROC (left) which is based on ratio independent TPR and FPR, the random baseline of a PRC (right) rises with a higher fraction of positive instances which can make it difficult to analyze the performance of different models. Models with AUC performing worse than the random baseline, like Model B on the left, usually need more investigation.

areas of machine learning with many model parameters. An alternative for such a case is a random parameter search. In a random search, multiple combinations with different parameter values are trained in parallel. The best combination is then selected from the random subset. Random search often finds that certain hyperparameter values are relatively flexible and can be excluded from further optimization. After an initial random search, hyperparameters can be further optimized by greedy Bayesian optimization, evolutionary algorithms, and, for some algorithms, by deriving parameter gradients.

Keep it simple in order to understand a predictions

In the 14th century, the English scholar William of Ockham formulated the principle: "*Numquam ponenda est pluralitas sine necessitate*" or "Plurality must never be posited without necessity" that is known today Ockham's razor. In the field of machine learning, this is adapted as "a model should be kept as simple as possible". While complexity naturally arises with large model sizes and longer and more complicated training processes, the main reason behind this is comprehensibility. Testing a model on a data set

may assure that the prediction is not entirely unfounded but does not tell why a model is giving a prediction. Especially in edge cases, when professional and computational prediction are diverging, this raises the question: is the prediction founded on a better understanding of the circumstances or due to an artifact? If this should be the case, can general rules be generated that explain the prediction? And if this is not the case, can the faulty logic in a machine learning model be found so that altering features can be corrected or the training process be adapted? Else, these problems may create loopholes that can lead to wrong decision-making or exploitation for coordinated attacks on a model that may be invisible to the uneducated user (Su *et al.*, 2019). Finding the features that lead to a prediction is called feature attribution. In a linear model, feature attribution is comparatively simple since the relative quantiles of the value of an instance are known for each feature, as well as the coefficients that are learned in the model. The same process is a lot more complicated for nonlinear models with many competing methods proposed like for decision trees (Lundberg *et al.*, 2020) and neural networks (Mundhenk *et al.*, 2020; Yang & Kim, 2019). The gained insight is used in the discovery of meta-features that could not be found in linear settings.

1.4 Unsupervised and Semi-supervised learning approaches

Machine learning does not have to rely on labeled instances. The field of unsupervised machine learning analyzes data set in order to recognize patterns that explain the underlying structure. In addition to finding hidden labels and dimensions, unsupervised learning algorithms are used for outlier or anomaly detection, recommender systems, and dimension reduction. The two main branches of unsupervised machine learning are, similar to classification and regression, clustering into groups and matrix factorization that detects modes of variation, although these two fields overlap. Clustering is a surrogate term for approaches that divide the instances into a finite set of groupings. These groups or clusters are learned by the algorithm based on the similarity between instances, trying to find groups where instances are more related to each other than to instances outside of the group. There are many different clustering algorithms based on the expected structure of the final clusters and the structure of the algorithm itself. To give an example, k-means clustering specifies the number of clusters at the beginning and which are then refined iteratively. In contrast, in hierarchical clustering, all instances are divided into two groups which are then recursively divided further into smaller groups until each subgroup only contains a single instance. Each cluster or grouping generated

by a clustering algorithm is described by a vector, either directly computed from the algorithm or via the mean per feature of all instances in that grouping. These clustering vectors are functionally equivalent to the modes of variation or dimensions in a matrix factorization. The main difference to the latter is that in a clustering approach each instance is matched to a single cluster while in matrix factorization, each instance is represented by a weight vector that describes the expression of all dimensions in this instance. Again, there are a lot of different algorithms that enforce certain properties on both weight and dimensions. Examples of these are orthogonality of dimensions in principle component analysis (PCA) and non-negativity of weights in non-negative matrix factorization (NMF). Unsupervised machine learning shares many of the steps and adaptions from supervised machine learning such as feature engineering and regularization.

Semi-supervised machine learning combines strengths from supervised and unsupervised machine learning approaches. It is applied if primary instances labels exist but are not available for all instances in the training set. It is based on the assumptions that the provided labels are representative of the true labels of all instances and that instances with similar features are likely to have the same label. The semi-supervised algorithm can then rely on labels for a supervised definition of the general scope of the prediction and later integrate a much bigger training set that can be analyzed unsupervised to estimate the background distribution of instances. Semi-supervised approaches are popular where instances are simple to generate but classes are expensive to assign accurately.

1.5 Additional concepts: Reinforcement, multi-task and transfer learning

For many problems, the evaluation of a solution is much simpler than the generation of a perfect solution. Examples of such problems are the result of a game of chess or the steering of a machine. In principle, it is possible for these problems to generate a number of solutions randomly, evaluate all, and in the end select by best performance. However, in many cases, this is not possible as it is the objective that generated solutions are good. Over the years, reinforcement learning has developed as a field of algorithms that train models that adapt to problems with a multitude of inputs in order to generate a good solution from different situations. The main premise in reinforcement learning is reinforcement, where the model repeatedly encounters similar situations from which

it generates action. That action is then evaluated and either positively or negatively "reinforces" the model to do the same action again in the next iteration. Reinforcement learning has been applied in many areas for playing games like chess or go and for performing interactions with the real world like maneuvering robotic arms or steering cars. A special case of reinforcement learning is the generative adversarial network (GAN). In a GAN setting, two machine learning models are trained simultaneously. The first, discriminative model can be any supervised or semi-supervised machine learning model that is trained to evaluate instances while minimizing a loss function. The second, generative model is generating new instances. While the initial distribution of these instances is derived from a normal training set, the adversarial part of this type of machine learning comes from the fact that the generative model is trained as well on the error rate of the first model, it has the objective to increase the number of classification mistakes. This is supposed to lead to the increased generation of particularly hard training cases that therefore get reinforced more than simple cases on the discriminative network. A typical application of a GAN is the discrimination between random computer-generated images and for example images of human faces (Karras *et al.*, 2020). In that case, the generative model is supposed to learn to generate more and more realistic synthetic face images that are hard to distinguish from real faces. Unfortunately, the usefulness of GANs is limited by the fact that those tend to concentrate on a small number of extreme cases while neglecting large parts of the overall input space (Arora *et al.*, 2017).

Multi-task learning

By design, machine learning algorithms optimize the problem they are given: classifying instances from the given training data set. If a model is trained to distinguish between tomato and cherry images, it will do just that: distinguish between tomatoes and cherries. Now the same training data set of tomatoes and cherries may be classified not by the type of fruit but by whether the fruit is ripe and edible. It may be the simplest to train two models for each tomato and cherry ripeness. However, predicting both properties in what is called a multi-task model, may be beneficial for covering certain rare cases that are covered sparsely in the tomato or cherry subsets of the entire data set. In this example, while color is usually a good indicator of ripeness there are strains of both tomatoes and cherries that are yellow when ripe. While this may be rare within the entire data set and may therefore be ignored or overfitted in a classifier of a single type of fruit, the combined data set contains more instances over a broader background which

may make it easier for the model to learn that color does not necessarily equate ripeness. And while training a separate model that only trains general fruit ripeness may cover this case, there are certainly still differences in how cherries and tomatoes ripen, and therefore important that the model still learns to distinguish different types of fruit by training to predict both properties jointly. Another example of this problem would be the classification of certain images by day and by night, where different features become important depending on the other classification results and a transfer of those features improves overall prediction.

Transfer learning

Related but not the same as sequential training is transfer learning. In transfer learning, parts of an already existing model are adapted for the training in another model. This can mean that the parameters of a trained model are used as initialization for a new model or even that entire components of the model are adapted. For example, layers of a deep neural network can be used as fixed feature processors on whose output the new model is trained. Reasons for using parts of or even entire preexisting models are either the amount of time that it takes to train a new model or the limited size of the training data set, which may not be big enough to train a fully regressed model without overfitting. Both scenarios are common in image classification where complex pre-trained models like ImageNet that have been trained on large image data set and can extract important features from these images are adapted for a new type of image prediction.

2 Variants in the human genome

Deoxyribonucleic acid (DNA) is the key molecule storing instructions to operate cellular life. DNA is composed of a double helix built from the four nucleotides adenine (A), cytosine (C), guanine (G), and thymine (T). The two strands of the double helix are composed of matching nucleotides, with A being paired with T on the opposite strand and G with C. Each of these pairs is one base pair (bp) in the DNA molecule. The encoded information serves as blueprints for thousands of molecules and specifies when to build which. In total, two sequences of more than 3 billion bp form the human genome, the collection of all human genetic information.

Each normal human cell nucleus contains 46 chromosomes, 2 copies each of the 22 autosomes as well as two gonosomes, either two copies of chromosome X in case of a female cell or one of X and one of Y in case of a male cell. In addition, a small mitochondrial genome is located in separate organelles outside of the nucleus, the mitochondria, which are found as multiple copies in each cell. Each chromosome (and the mitochondrial genome) contains many genes. These genes are translated into proteins and ribonucleic acids (RNA), specialized molecules that serve special functions like building more molecules, harvesting energy, or protecting the cell from external agents.

This chapter serves as a brief introduction to the genome and genetic variants. It introduces why the genome sequence is different for every individual and what consequences these differences may have. It also describes different types of genetic variants and how those are discovered. While the details on how variants cause disease are described in the next chapter, I hope to motivate the steps necessary for variant interpretation.

2.1 Definitions

The information of a segment of DNA, potentially controlling all kinds of cellular processes, is encoded in its nucleotide sequence. Hence, if the sequence of two molecules of DNA is identical, these two molecules could be assumed to functionally have the same effect. Any nucleotide in the DNA sequence has a position and the sequence is oriented. If a DNA molecule is altered by exchanging some nucleotides with others, the change in sequence is called a mutation. Mutations have a direction (i.e in temporal order) from one sequence to another. They are formally described by the properties position, original (reference) sequence, and new (alternative) sequence. When there is a difference between two sequences but the direction is unknown or simply ignored, the

difference may be called variant, with the position having genetic variation. Since variants are usually described in comparison to a reference, they are characterized by the same three properties as mutations. In practice, the term mutation is often replaced by the term variant. Frequently, the type of variant is further specified by attributes such as pathogenic or benign (Richards *et al.*, 2015).

Like each nucleotide has a position in a DNA molecule, the genomic locus describes the position of a sequence on a chromosome in the genome. A sequence version at a particular locus in the genome is called an allele. Different alleles usually share a common sequence origin but differ in one or more variants.

The term for having two copies is diploid, while a single copy genome is haploid. Most chromosomes exist diploid in a normal human cell nucleus. Exceptions of this rule are the sex-determining gonosomes X and Y: While X is diploid in females, it is haploid in males as there is also one copy of chromosome Y. Similar to chromosomes, variants (or alleles) can also exist in one or two copies. If found only on one chromosome, the variant is heterozygous. In contrast, if both chromosome copies have the same allele, they are homozygous.

In principle, the two copies of a diploid chromosome are independent of each other. While there are some interactions between chromosomes and certain types of mutation like recombination can lead to exchanges of segments between chromosomes, the respective sequences are independent of each other. Each of the two copies is referred to as a haplotype, with the paternal haplotype being inherited from the father and the maternal from the mother. Multiple variants of the same gene or chromosome can be on the same haplotype or not. Variants on the same haplotype are called in phase or phased. Phasing becomes important when two variants would both impact the function of a gene: If both variants are in phase, only one copy of the gene is affected and the other may rescue its function.

The collection of all genomic information including all variants in a cell or organism is the genotype. The genotype includes information about heterozygous and homozygous copies of each allele, and is descriptive either for the entire genome or limited to a certain locus. Each genotype is linked to a particular phenotype, the expression of observable traits in the organism with that genomic information. The phenotype can be but does not have to be dependent on the genotype. A variant responsible for or contributing to a certain phenotype, for example a genetic disease, is called causal variant.

2.2 Determining variants in the DNA sequence

DNA sequencing is the process that reads the order of the four nucleotides A, C, G, and T in a DNA molecule. Due to the complementary base pairing of A with T and C with G, sequencing is necessary only for a single strand of DNA. The nucleotides along the strand are read out one after the other, as they are molecularly bound and oriented in the nucleotide chain.

Since continuous stretches of DNA, i.e. entire chromosomes, are millions of nucleotides long, it is currently technologically impossible to consecutively read the entire sequence. Instead, sequencing is performed from DNA fragments. Each of these fragments, colloquial known as reads, covers a tiny fraction of the total sequence. The original sequence is later reconstructed by integrating the information of all reads from a sequencing run. Read properties like lengths and error rate, which are important for that process, are sequencing technology dependent.

Since the beginnings in the 1970s, numerous technologies have been developed for DNA sequencing. All of these are specialized in how a DNA sample has to be prepared and how the actual sequencing is performed. Today, the actual sequencing step is run through a dedicated machine, called the sequencer, that can sequence hundreds to millions of DNA reads in parallel. In contrast to older and more laborious methods, these high-throughput techniques are called next-generation sequencing (NGS).

The most broadly applied, NGS technology is bridge amplification coupled with sequencing by synthesis (SBS), as commercialized by the company Solexa (now Illumina). Bridge amplification is used to locally amplify a DNA molecule on a surface so that many copies of the same molecule are located in one area of the sequencing chip. The basic principle of SBS is the synthesis of the matching second strand of the DNA molecule. The insertion of the complementary nucleotides in all clonal copies of the molecule is detected and monitored. As different fluorophore colors are linked to the integration of each nucleotide, the sequence of the DNA molecule is decoded. With read lengths of up to 300 nucleotides, SBS is a short-read sequencing technology.

In recent years, short-read sequencing has been complemented by single-molecule and long-read technologies, as being developed by the companies Pacific Biosciences and Oxford Nanopore Technologies. Long-read technologies provide reads of up to mega (10^9) bases in lengths (Payne *et al.*, 2019). Ultra-long reads are used to connect distant sequences, providing additional information about a single haplotype (Beyter *et al.*, 2021) and enable more complete genome reference, including highly repetitive regions

that could not be resolved with short reads (Nurk *et al.*, 2021). However, available technologies show higher and partially context-dependent error rates than SBS (Amarasinghe *et al.*, 2020) and are, as of 2020, more expensive per base pair of sequenced DNA (Logsdon *et al.*, 2020).

The genome of a human, as well as any other organism, can be sequenced to different levels of quantity or depth. The term coverage describes how often each position is included in (or covered by) single reads on average. For an entire human genome with its 3 billion bp, this means that at least 10 million reads of 300 bp are necessary to reach an average 10-fold coverage of the genome. While whole-genome sequencing (WGS) is done routinely nowadays, especially so-called non-coding regions are frequently not analyzed to save cost and time. Instead, whole-exome sequencing (WES) is limited to the exome, the coding exons, and nearby untranslated regions of the genome. The DNA library that is used for WES is generated by enriching coding DNA regions via targeted sequence capture (Ng *et al.*, 2009). For analyses of a single disease, where only a few known genes are of interest, enrichment can be further limited to what is known as a gene panel.

In recent years, a huge number of protocols have been established that extend upon the concept of DNA sequencing. These enable the sequencing of different types of RNA, determine DNA methylation, mark the binding of different molecules to DNA or interrogate the proximity of DNA segments in the nucleus. A recent hype has developed around the field of single cell genomics that associates each sequencing read with a unique cell. Sequencing many cells in parallel enables deeper insights into the regulatory trajectories that cells progress through.

From sequencing to calling variants

DNA sequencing of a sample does not provide a full genome sequence but rather a high number of individual DNA reads. Depending on the application, there are two orthogonal approaches to obtain the genome sequence: Either de novo assembly, only based on the DNA reads, or with help of a reference genome similar to the sample.

When no reference genome is available, is too distantly related, or missing important segments, a new genome has to be assembled by connecting reads to a single continuous sequence. This is performed based on overlaps at opposite read-ends, which are merged into one longer sequence. Ideally, many reads overlap continuously, so that large sequence blocks that cover large genomic areas are obtained. Due to mismatches and non-unique

nucleotide k-mers, it is nearly impossible to decode entire chromosomes in one sequence based on overlaps alone. Current algorithms for genome assembly resolve ambiguities via a type of directed graph, the De Bruijn graph (Simpson & Pop, 2015), by finding a path through every node in the graph. A major challenge in this regard are repetitive regions where the same sequence pattern appears multiple times in close succession. Reoccurring patterns lead to loops in the graph that could be explained by a virtually unlimited number of possible sequences. To solve the sequence in these repeat regions, long-read technologies are popular where the reads cover the entire repeat.

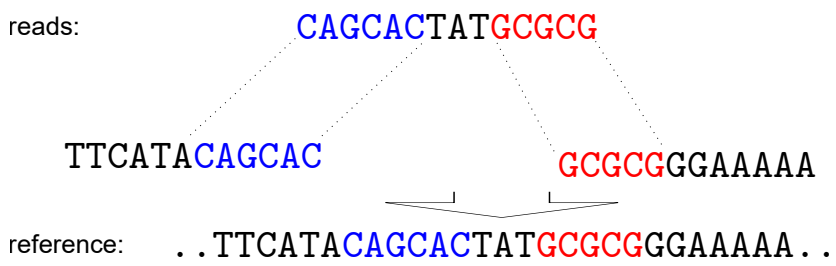

Figure 7: **Concept of sequence assembly.** Aligning segments of multiple reads are merged into one larger sequence, and finally integrated in building the reference genome. In reality, the toy-example above is made more complicated by read errors and repetitive regions where multiple, conflicting overlaps are possible.

While single genes were assembled much earlier, the Human Genome Project took hundreds of scientists from institutions all over the world years to sequence the first human genome. In 2003, the formed Genome Reference Consortium released the first version of what is known today as the human reference genome. That reference is not one particular human genome but an idealized representation of a genome for the entire species. GRCh38, the latest human reference genome, has a total length of about 3.1 billion nucleotides but includes several unresolved gaps. These gaps are caused by complex regions like the chromosome centromeres that are rich in near-identical sequence repeats, so that about 8% of that sequence remains unresolved in GRCh38. Only recently, the Telomere-to-Telomere consortium is decoding the full sequences of the last unknown segments and providing a genome sequence without gaps (Nurk *et al.*, 2021).

Genome assembly is expensive due to the high read lengths and number of reads that are necessary to obtain a high assembly quality. Instead, most analyses rely on a reference like the human reference genome and look for differences in their data in comparison to that of the reference. In a step called mapping, the sequenced reads are

aligned to the reference (Fig. 8). Mapping takes into account that not all reads match perfectly with the reference. Mismatches may be caused due to differences between sample and reference, but also due to the reference missing that particular sequence, high sequence divergence, sequencing errors, or sample contamination.

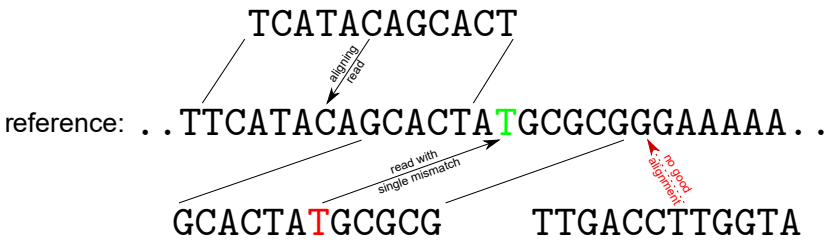

Figure 8: **Read mapping.** Sequenced DNA reads are matched to the already existing reference sequence. Mismatches in the mapping between read and reference may be due a difference in the sampled sequence, sequencing errors or because the reads true alignment is to another location of the genome.

Variants are called once the reads are mapped to the reference. Variant calling means going through the alignments and determining sequence differences from the reference that are shared between multiple reads. Strong quality control is needed to screen the found differences to minimize false positive results due to sequencing errors and misaligned reads. All variants that pass these checks are collected and characterized in terms of sequence difference, position, quality metrics, and number of observed reads to distinguish heterozygous from homozygous variants. To obtain additional information about haplotypes, variants located in close proximity may be phased by testing if they appear on the same reads.

Structural types of genetic variants

The most frequent type of difference between two genomic sequences is the single nucleotide variant (SNV): the exchange of one nucleotide in sequence A by one of the three others in sequence B. Based on the structure of the nucleobase molecule, nucleotides are divided into purines (Adenine and Guanine) and pyrimidines (Cytosine and Thymine). As these structures influence the DNA secondary structure, SNVs are divided into transversions, where a purine nucleotide is exchanged to a pyrimidine nucleotide or vice versa, and transitions, where the basic purine (or pyrimidine) structure remains in place. By combining the four DNA nucleotides, there are 12 possible SNVs, three of which are possible at each position in a sequence. In addition, the unaffected adjacent nucleotides,

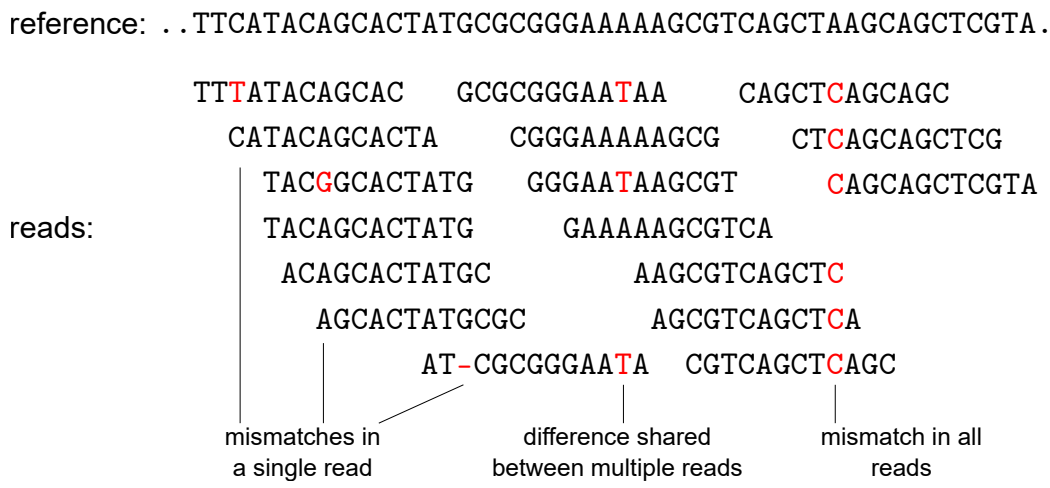

Figure 9: **Variant calling.** Variants are called from the difference between aligned reads and the reference sequence. Mismatches found only in a minority of reads are likely to be due to sequencing errors or misalignment. The task of the variant caller is to find variants that are supported by multiple reads. Depending on the fraction of reads a variant is observed in, there are heterozygous variants that are only found on one allele and homozygous variants that appear in all reads.

e.g. +1 and -1 of the affected position, are often considered when describing a SNV as those play an important role in how variants arise (see next chapter). When multiple SNVs appear in close succession on the same haplotype, they may be considered as a mutational hotspot.

In comparison to the limited number of possible SNVs, there is an infinite amount of possible larger variants (Fig. 10). These include the addition of novel DNA sequence in an insertion and the removal of DNA sequence in a deletion. By definition, insertions and deletions, often summarized as InDels, are less than 50 bp long.

Sequence alterations larger than 50 bp are called structural variants (SVs). There is no length limit for SVs, including relocation of entire chromosome arms, though most frequent human SVs are shorter than 100,000 bp long (Collins *et al.*, 2020). SVs, specifically large deletions and insertions, can be further classified according to where the inserted sequence comes from or the deleted sequence goes. Accordingly, there are dup- and multiplication of an existing sequence, translocations where a segment of DNA is moved to another genomic locus, and inversions where a segment is reversed in its orientation. These large-scale changes may be accompanied by multiple smaller sequence changes like SNVs (or other SVs) in the sequence, further complicating the discovery

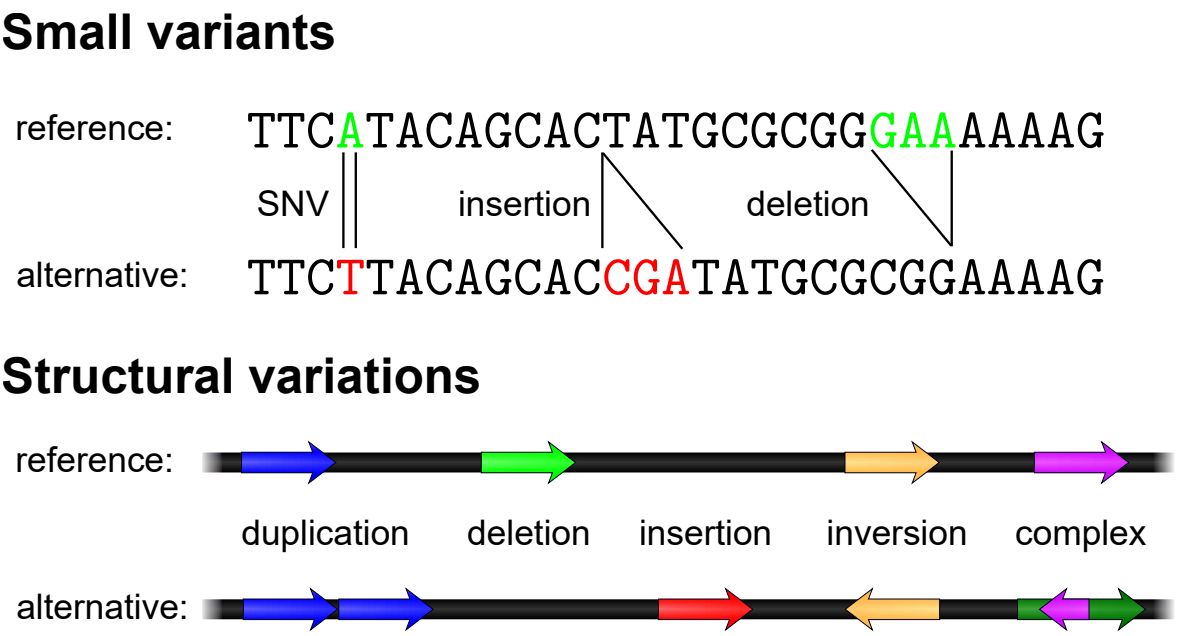

Figure 10: **Types of small and large variants.** All variants are identified as sequence differences of a sample (alternative) compared to the reference sequence. Defined as small variants are SNVs as well as insertions and deletions (InDels) that affect less than 50 bp of DNA sequence. Larger variants are called structural variants (SVs).

and classification.

In principle, any type of variant may have a phenotypic effect. Since most variants are SNVs and short InDels with less than three de novo SVs discovered in an average individual (Abel *et al.*, 2020), many studies primarily focus on those smaller variant types. This is also influenced by the fact that processes like variant calling heavily depend on the type of variant: While SNVs will mostly lead to base mismatches in the aligned sequence read, longer insertions may align a read to different genomic regions. Larger variants remain important nonetheless. They were found causal for many diseases (Mitelman *et al.*, 2007; Sanchis-Juan *et al.*, 2018) and are considered to be more likely pathogenic per variant (Ganel *et al.*, 2016), mostly due to the larger absolute change and more affected genomic sequence (Sudmant *et al.*, 2015).

2.3 Sources of variants

Most variants in a genome are shared with a parent (Fig. 11). They may have appeared first in a distant ancestor and have been passed on ever since. All of these variants have

accumulated since the last common ancestor of all humans, a hypothetical person from which the genome sequences of all humans living today are inherited. The last common ancestor has been estimated for the mitochondrial genome and chromosome Y in the range of multiple hundred thousand years ago (Fu *et al.*, 2013; Poznik *et al.*, 2013). Since chromosome Y and the mitochondrial DNA are the parts of the genome where few recombinations happen, increasing the likelihood that large stretches from a single genome are inherited by all individuals in a population (Jobling & Tyler-Smith, 2017), the last common ancestor of other regions of the genome and variants within those may be older than that.

Every variant in a genome has appeared once de novo in one individual through mutation of the germline at that position and has since been inherited by that person's progeny. Trio sequencing studies, that compare the genome of a child to those of its parents, have found a median of 60 to 70 de novo variants, variants that are limited to the child (The 1000 Genomes Project Consortium, 2011; Kong *et al.*, 2012; Kaplanis *et al.*, 2021). Some de novo variants are shared between siblings and derive from the same parental gamete lineage where the mutation appeared (Jónsson *et al.*, 2018). The number of these mutation increases with parental age (Jónsson *et al.*, 2017). More de novo variants arise from the first cell stages of the developing baby, though later mutations won't affect most cells of the developing organism.

There are two main categories to classify sequence alterations by cause: spontaneous mutations that appear due to structural damage to the DNA molecule and copying errors that occur when the DNA is replicated (Ségurel *et al.*, 2014). Endogenous causes for spontaneous mutations include the formation of pyrimidine dimers and hydrolytic deamination of cytosines (Lindahl, 1993). If not repaired, the altered nucleotides are not recognized properly in the next DNA replication and a mutation manifests in one of the daughter cells (Maki, 2002). Similarly, exogenous mutagens like radiation and reactive oxygen species can induce spontaneous mutations by causing reactions of the DNA nucleotides or breaking one or both DNA strands (Ségurel *et al.*, 2014). Additionally, replication errors occur when the DNA polymerase incorporates the wrong nucleotide in the newly synthesized strand, a mistake that happens in about 1 in 10^4 to 10^5 bp (Echols & Goodman, 1991) but is reduced to about 1 in 10^9 bp by proofreading exonuclease domains in the DNA polymerases (Schmitt *et al.*, 2009; Korona *et al.*, 2011).

In most cases, the described DNA damage affects only one of the two strands of the DNA double helix. These premutations only become permanent after the next

```
reference        ..TTCATACAGC--ACTATGCGCGGGAAAAAGCGTCAGCTAAGCAGC..

maternal  m1:..TTCATACAGC--ACTATGCGCGGTAAGAAGCGTCAGCTAAGCAGC..
genome    m2:..TTCCTACAGC--ACTATGCCCGGGAAAAAGCGTCAGCTAAGCAGC..

paternal  p1:..TTCATAGAGCTGACTATGCCCGGTAAAAAGCGTCAGCTAAGCAGC..
genome    p2:..TCCATAGAGC--ACTATGCGCGGGAAGAAGCGTCAGCTAAGCAGC..

child     m1:..TTCATACAGC--ACTATGCGCGGTAAGAAGCGTCAGCTA--CAGC..
genome    p2:..TCCATAGAGC--ACTATGCGCGGGAAGAAGCGTCCGCTAAGCAGC..
                 | |   |                 |         |       |   |
          heterozygous variants  homozygous variants  genotype of child can be  de novo SNV  de novo deletion
          may be inherited       are always inherited  different from both parents

child     m1:..TTCATACAGC--ACTATGCCCGGGAAGAAGCGTCAGCTA--CAGC..
tissue    p2:..TCCATACAGC--ACTATGCGCAGGAAGAAGCGTCCGCTAAGCAGC..
                                     |
                                  somatic SNV
```

Figure 11: **Inherited, de novo, and somatic variants.** Most variants in every individual genome are shared with one of the parents. In contrast, de novo variants are not observed in the genome of the parents. Somatic variants are those variants that are not found in all cells of the organism. Depending on when the mutation occurred, they are observed in only one or multiple tissues.

cell division (Maki, 2002). Molecular repair pathways exist to correct premutations that appear as mismatches between DNA strands. Repair mechanisms are diverse and include mismatch repair such as the proofreading during DNA replication, nucleotide excision repair and base excision repair that respond when one or multiple nucleotides are damaged and need to be replaced, and the repair of double-strand breaks via homologous recombination, matching the sequence to a sister chromatid, or via non-homologous end joining (NHEJ) (Iyama & Wilson, 2013). Mistakes that happen in DNA repair or that are a result of incomplete repair during NHEJ are an additional source of mutations (Lieber, 2010). Variants in the associated repair genes can reduce or entirely prevent the function of repair mechanisms so that more premutations will proliferate. While heterozygous repair gene defects can increase the number of mutations (Sharma *et al.*, 2020), especially homozygous defects lead to very high mutation rates and are associated with somatic (Roberts & Gordenin, 2014) and germline hypermutation events (Kaplanis *et al.*, 2021).

The likelihood for a process to cause a mutation depends on the chromatin structure, the DNA sequence, and the type of mutation (Sabarinathan *et al.*, 2016; Mao & Wyrick, 2019). Some types of DNA damage like nucleotide deamination can only happen at certain nucleotides. Accordingly, the number of mutations attributed to each process follows characteristic nucleotide distributions. These process characteristics are found stochastically in the form of mutational signatures in genome-wide analyses of mutation counts (Alexandrov *et al.*, 2013). Originally discovered in cancer studies, by now more than 100 different mutational signatures have been established and are attributed to many different sources, from repair enzyme defects to chemotherapeutic agents (Phillips, 2018; Tate *et al.*, 2019; Alexandrov *et al.*, 2020). As such, mutational signatures have also been established for de novo mutations to describe global mutation distributions and hypermutation events (Kaplanis *et al.*, 2021).

Somatic variants and cancer

Germline variants exist in a genome since the fusion of maternal and paternal germ cells and hence are shared between all cells in the organism. Somatic variants are mutations that appear after the first cell division of the fertilized oocyte and are thus not found in all cells. By definition, somatic variants exclude mutations in the germline, i.e. reproductive tissues, therefore only affect the individual and are not inherited by offspring. Over a lifetime, organisms continuously accumulate somatic variants in all body parts, with increased numbers found in environmentally exposed and highly proliferating tissues (Yizhak *et al.*, 2019).

By disrupting cellular processes, the increasing number of variants is thought to be one of the reasons for aging (Campisi & Vijg, 2009). High mutation rates have been found to lead to severe damage of organ function, premature aging and are linked to neurodegeneration and cancer (Kennedy *et al.*, 2012). Somatic variants located in genes that control processes like cell proliferation or cell death may disrupt the regulation of those processes, lead to cellular aberrations and ultimately cancer. While not all cancer cases start from a genetic cause (Yu *et al.*, 2014; Ostrander *et al.*, 2016), mutations are thought to be necessary for the progression of every tumor (Morjaria, 2021). A typical tumor genome contains 4 or 5 driver mutations, with most other somatic variants currently believed to be passenger variants (Campbell *et al.*, 2020). The driver mutations enable faster growth, are responsible for evasion of the immune system, and facilitate resistance to chemotherapy (Stratton *et al.*, 2009). While not directly responsible for the

unregulated growth, deleterious passenger mutation may contribute to the optimization of the tumors growth strategy (McFarland *et al.*, 2013). Not all variants that increase the likelihood of a tumor or drive progression are somatic and may be inherited over multiple generations (Brown *et al.*, 2020). Often located in DNA repair genes, these cancer syndrome variants are linked to increased basal mutation rates that eventually lead to unlimited cell proliferation and cancer (Kaplanis *et al.*, 2021).

Population sequencing and standing variation

Since the initial release of the human reference genome, various consortia have started to sequence many human genomes in order to map variants across the world. While initial human population whole-genome sequencing projects like 1000 Genomes and HapMap already included more than 1,000 people, decreases in sequencing costs enable recent projects like gnomAD, TopMed, UK10K to study 100,000 and more individual genomes.

Standing variation is the incidence of more than one allele in a genomic locus. It is thought to be a major source of local adaption as variants that increase evolutionary fitness are positively selected in a novel environment (Barrett & Schluter, 2008). Examples of local adoption are variants in the gene HBB that are most frequently observed in tropical regions of Africa and, when heterozygous, provide resistance to malaria (Rees *et al.*, 2010). Variants in the genes MTHFR and EPAS1 frequently observed in Tibetans are thought to enable better oxygen uptake at higher altitudes (Yang *et al.*, 2017a). Not all local differences are due to selection though but can also appear because of genetic drift, changes in allele frequency that appear as a result of random mating (Rees *et al.*, 2020).

In addition to trait associations, studies of standing variation and genetic drift have provided new insights into the historic distribution and interactions of human populations, both on a global scale (Elhaik *et al.*, 2014) as well within countries (Leslie *et al.*, 2015). Accordingly, those variants can be used to geographically map humans in the world based on relatives and ancestries. Some standing variation is shared even with the genomes of Neanderthals (Green *et al.*, 2010) and Denisovans (Reich *et al.*, 2010). A small excess of this allele sharing has led to the hypothesis that modern humans have had admixture with those extinct human lineages (Lohse & Frantz, 2014), with some variants in every non-African human being passed down from shared Neanderthal or Denisovan ancestors. These variants are an important contribution to the diversity and local adaptation of modern humans (Dannemann & Kelso, 2017).

2.4 Genome-wide association studies and polygenic risk scores

Most variants in a human genome are not unique to the single individual but shared with other humans. Large population sequencing studies such as gnomAD (Karczewski *et al.*, 2020) and BRAVO (Taliun *et al.*, 2021) have found hundreds of millions of variants. As of gnomAD version 2.1, 25 million SNVs and 6.7 million InDels have allele frequencies greater than 0.1%. Frequent occurrence of variants enables statistical analyses, called genome-wide association studies (GWAS), that integrate genotypes over hundreds of genomes in order to link the significant enrichment of certain variants to measured phenotypes (Hirschhorn & Daly, 2005).

Fine mapping of significant associations between a trait and a single causal variant is impaired by linkage disequilibrium (LD) as phased variants are not found independently (Slatkin, 2008). The association of one or multiple causal variants may lead to enrichment in nearby, linked passenger mutations that do not have any effect themselves. While LD often complicates pin-pointing an effect to a single variant, numerous GWAS hits have been validated in independent experiments or with high resolution and are considered confirmed trait loci (Schaid *et al.*, 2018). GWAS have been used to study thousands of different traits and found numerous links to disease or phenotype (Visscher *et al.*, 2017; Buniello *et al.*, 2019). Based on these findings, multiple medications have been developed that are prescribed based on findings of certain genetic variants (Szustakowski *et al.*, 2020).

As common variants, most GWAS hits typically have relatively small effect sizes with larger trait differences being caused by multiple loci (Tam *et al.*, 2019). However, knowledge about multiple loci that influence the same phenotype can be combined to make stronger predictions about the trait from the human genome. Polygenic risk scores (PRS) estimate the cumulative effect of all common variants in a genome on a certain trait (Dudbridge, 2013). While PRS have successfully been applied to predict lifetime risks of disease (Lewis & Vassos, 2020), recent studies have also shown that the proportion of total variance explained by PRS depends on ethnicity and social-economic status, raising questions about sampling bias in the sequenced human genomes (Duncan *et al.*, 2019; Mostafavi *et al.*, 2020).

2.5 Rare Mendelian disorders

While the combined effect of multiple GWAS hits can be strong, most common variants have relatively small effect sizes and explain only a fraction of total variation (Conrad *et al.*, 2010; Lupski *et al.*, 2011; Marouli *et al.*, 2017). As evolutionary pressure selects against variants with negative effect sizes and for variants with positive effect sizes, functional variants are generally depleted among common and hence enriched among rare variants (Marth *et al.*, 2011). Large negative effects are often seen for de novo mutations (Acuna-Hidalgo *et al.*, 2016; Deciphering Developmental Disorders Study, 2017). As de novo mutations may appear anywhere in the genome, there is a large variety of phenotypes, with over 15,000 predicted disease genes (Cooper *et al.*, 2010) and over 6,000 described genetic disorders (Amberger *et al.*, 2019). Many disorders have been described for a small number or even just one individual and are hence known as rare genetic diseases. In the past, rare disease mechanisms were studied by screening many different biological pathways and biomarkers in order to find differentially expressed genes and modified proteins (Rodenburg, 2018). Patients with a similar phenotype were summarized into study groups to find a common etiopathogenesis, the shared cause and development of an abnormal condition. High-throughput sequencing technologies have enabled trio sequencing (Need *et al.*, 2012), where the genomic sequence of the child is screened for variants that are not found in the genomes of either of its parents (see Fig. 11). Pooling rare genetic disease patients not just by phenotype but by variants in the same genes improves the discovery of previously unknown rare disease genes (Wright *et al.*, 2018; Kaplanis *et al.*, 2020), and supports the development of molecular treatments based on response in those groups (Pogue *et al.*, 2018).

Many rare diseases are monogenic, meaning that they are caused by variants in a single gene. The large effect size of a single causal variant, that disrupts the proper function of the involved gene, may cause a malformation. Monogenic diseases are also called Mendelian disorders, named after the genetic inheritance rules first discovered by Gregor Mendel. This distinguishes Mendelian from complex disorders that are caused by a combination of environmental and genetic factors. Despite the name, many Mendelian diseases lead to early death or infertility and are inherited less frequently as carriers do not have any progeny (Mazzucato *et al.*, 2014). Instead, the causal variant may appear de novo and is only observed in the germline of the child (Deciphering Developmental Disorders Study, 2017).

2.6 The genetic variant interpretation challenge

The number of genomic differences of any human's personal genome in comparison to the reference genome is in the order of multiple millions in total (The 1000 Genomes Project Consortium, 2015). This leads to a single-digit number of variants per thousand bp of DNA, unevenly distributed over the whole genome (Rogozin & Pavlov, 2003). The challenge in interpreting any single human genome is identifying the subset of causal variants, that is responsible for a particular phenotype.

To cause a change in a macroscopic trait, a variant has to influence a molecular process that is causing the altered phenotype. Based on the known difference to the annotated genomic sequence, it is possible to predict those variant effects on different molecular processes (Fig. 12). For example, the effect of a coding variant may be predicted as the integration of a different amino acid depending on the nucleotide triplet in the mRNA. Sequence predictions are very accurate, with possible predictions ranging from a synonymous variant that has no consequence to the amino acid sequence to a non-sense variant that causes the immediate termination of the protein's amino acid sequence. Other molecular processes like RNA splicing or transcription factor binding are harder to assess, although progress in deep learning has led to the development of tools that predict variant effects with previously unknown accuracy (Zhou & Troyanskaya, 2015; Jaganathan *et al.*, 2019; Cheung *et al.*, 2019; Avsec *et al.*, 2021).

Trait altering variants may be divided into two general molecular effect categories: gain of function and loss of function. The terms refer to the impact on molecular processes within a single cell like protein binding or formation of splicing isoforms. In that sense, loss of function is the disruption of a process, e.g. two proteins stop interacting or an existing site loses any splicing activity. Gain of function is then the formation of a new process, e.g. a novel interaction or splice site. The reason for this distinction is different predictability. As an existing activity is linked to a genomic sequence, mutations to that locus will very likely have a disrupting impact on that activity. On the contrary, a new activity may be one of many possible alternatives, e.g the novel protein interaction may be with any of thousands of proteins. Further, the existing activity is likely to play a certain role in the cell as else it would have been lost over time, while a new one may have no further consequences aside from the activity itself. Therefore, it is not surprising that concordance of variant consequence interpretations is higher for stop-gain mutation, that are typically loss of function, than for splice site or protein sequence changes, that may be loss or gain (Yang *et al.*, 2017b).

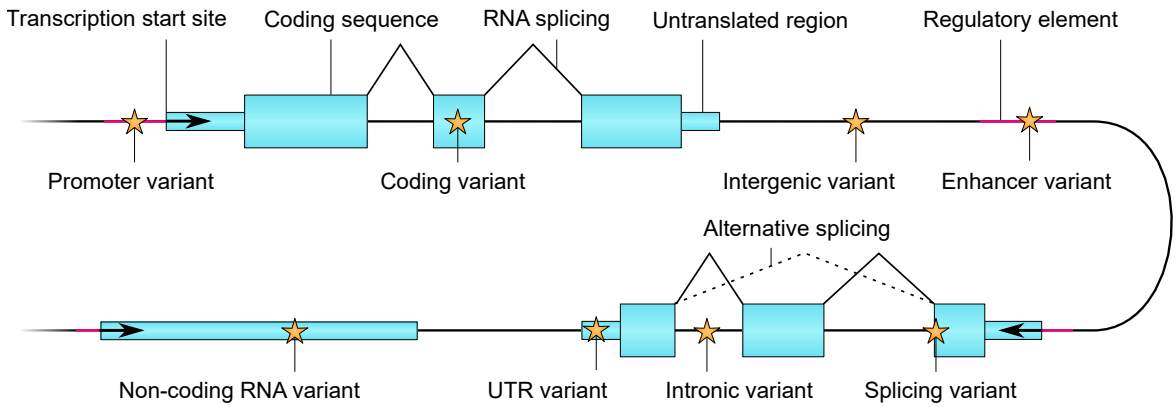

Figure 12: **Consequences of a variant.** Depending on the location in the genome, variants can influence different cellular processes. While some like promoter, enhancer or UTR variants are mostly effecting the expression of the association gene, splicing and coding variants may alter the structure and therefore function of the translated protein. A majority of randomly sampled variants, including most intronic and intergenic variants, will not have any affect on gene expression.

Properties of a causal variant

A variant may not have the same effect in all humans. For some carriers, a variant may lead to a genetic disease while other carriers are not affected. Penetrance is the probability that a person has a disease given a certain genotype. For example, a variant has a 70% penetrance regarding a disease if 7 in 10 individuals that carry the variant develop clinical symptoms of that disease. A variant has complete penetrance when all carriers of the variant are affected, otherwise the variant has incomplete penetrance.

Known since the famous pea experiments of Gregor Mendel, genetic traits may be inherited dominant or recessive. Dominant means that a single copy inherited from either parent is sufficient to cause the linked phenotype. In contrast, a recessive allele needs to be on both haplotypes to have the same effect. While for a dominant trait, at least one parent has to have the same trait, recessive traits may not appear in any ancestor, although both parents carry one copy of the causal allele. Dominant and recessive are mostly defined for alleles rather than specific variants. A person may have two alleles of a recessive trait and hence express the linked phenotype, despite the variants on the two alleles not being identical.

Some traits are neither entirely dominant nor recessive but on an intermediate level between those. These may be explained by genes where the two copies of an allele can

in principle compensate for each other but multiple variants may cause the effect. In this case, both a homozygous genotype as well as a heterozygous genotype with a second variant on the complementary haplotype may cause the phenotype.

Incomplete penetrance of a single variant can be caused by environmental factors, but may also be caused by interaction with other genetic variants (Cooper *et al.*, 2013). For example, a frameshift mutation alone has strong consequences for the expressed amino acid sequence of a gene. If there is a second frameshift in the same gene and the shifts of the two cancel out, the effect on the protein may be rescued. Similarly, the effect of any SNV may be masked by an upstream frameshift, or a splicing change may only be caused by some variant combinations (Baeza-Centurion *et al.*, 2019). Compensation or masking of variant effects by variants in other genes is called epistasis.

Terms used to describe variants can be different between medical and population genetics perspectives. If a variant is causing a negative effect, the medical field prefers the term pathogenic while the latter uses the term deleterious. While both words are often used synonymous, here they have distinct meanings: Variants are pathogenic if they are an actual disadvantage for the carrier by causing an unwanted phenotype. When a variant is deleterious, it may be pathogenic or it may not have any effect on the carrier, often due to incomplete penetrance. Nevertheless, the deleterious variant is disadvantageous from an evolutionary perspective because it may cause a negative effect in another situation, e.g. when being homozygous or through epistasis with other variants. Over multigenerational time scales, there is selection against deleterious variants. When variants do not have an effect on selection or phenotype, the medical term benign and the population genetics term neutral are used synonymously.

Ranking and validating causal variants

Identifying a strong molecular variant effect does not prove a link to a change in phenotype. Even stop-gain variants, of which on average between 1 and 2 are found in every healthy human genome, may not have a strong negative effect, with those variants being enriched in non-essential genes that are thought to have no impact on the carrier (Karczewski *et al.*, 2020). Conservation scores that track the number of mutations over long evolutionary time scales, as well as regional variant frequency indices, are tools for assessing the robustness of genes, non-coding regions, or parts thereof to novel variants. Variant effect scoring methods, introduced in detail in the next chapter, combine different sources of variant effect prediction in a single metric. The resulting score ap-

proximates variant deleteriousness and is used to prioritize variants by the likelihood of a disease-causing effect.

The highest scoring variants are finally investigated in detail to validate the link to the phenotype of concern and characterize a possible mechanism of action (Richards *et al.*, 2015), knowledge that can later be used to derive treatment options and actively influence the phenotype. Relatively simple functional assays like RNA-Seq, metabolome profiling, and analyses of protein expression are used to independently validate the variant effects (Rodenburg, 2018). Findings from many different patients and their medical data may be integrated in order to support the discovery of disease mechanisms based on statistical data (Kaplanis *et al.*, 2020). If these are not sufficient or to explore treatment options further, a more in-depth study including animal models, patient-derived or human-cell-line culture experiments may be necessary (Rodenburg, 2018). Best practices further call for validation of all causal variants by orthogonal sequencing technologies (Rehm *et al.*, 2013; Marshall *et al.*, 2020). Most of these validation experiments are laborious and expensive. It is therefore critical that ranking of all genomic variants is done comprehensively as to prioritize the validation of variants with a high likelihood of having an effect (Richards *et al.*, 2015).

3 In silico evaluation of genetic variants

Based on prior biological knowledge, an informed user can evaluate if a genomic variant is likely to cause disease. The used information about genes, sequence modifications, or evolutionary history of the sequence at a particular location in the genome is referred to as genome annotations. Single annotations are often not sufficient for a full evaluation, as even strong signals of pathogenicity like gain of a stop codon can turn out neutral (Rausell *et al.*, 2020). This makes variant evaluation laborious: hundreds of data annotations are available for every single variant and have to be put into relative context. Accordingly, there has been a long-time trend in computational biology to automate this task in a process called variant effect scoring. A variant effect score collects different annotations about a variant, evaluates the severity of each of those, and eventually generates a score that predicts the likelihood of a variant effect. In this sense, the score acts as a ranking mechanism that prioritizes variants that are likely to cause a change in phenotype over variants that are likely to be neutral. The goal is to evaluate all variants in a genome within a few hours at most. In this chapter, I review the general methodology of variant effect scoring methods, catalog variant annotations that can be used to predict variant pathogenicity, and give an overview of existing solutions to score genomic variants genome-wide. The chapter concludes with a detailed introduction to the variant effect score CADD.

3.1 Variant effect scoring

There are many different terminologies for the scoring, prioritizing, and ranking of genetic variants that are not standardized between different scientific publications. In the following, variant effect score describes a numeric value that is assigned to a genetic variant to assess its potential to have an effect. A variant effect score can predict a single effect like the disruption of RNA splicing or be rather generalized and summarize a multitude of different effects in a single metric that predicts pathogenicity. The score itself may be used to rank or prioritize variants. Variant prioritization itself may depend entirely on a variant effect score or may integrate additional, case-specific information. With that, prioritization may restrict variants to genomic locations within a set of genes that have been previously found to cause a similar phenotype, or separately handle previously classified variants.

The noun "score" here describes both the variant effect score of the single variant (e.g.

"Variant V_{42} has a score of 0.9") as well as the totality of all variant-value assignments (e.g. "The score has a range from 0 to 1"). Variant effect scores are the output of applying a variant effect model to genomic variants (Fig. 13). The "model" receives a genomic variant as input and generates the score based on information about the variant. The model has been developed using a particular variant scoring method based on prior knowledge about the metric the score is based on. The "method" specifies the overall model settings and the algorithm that determines the model parameters. In that sense, each model is an instance of the variant scoring method of which there can be multiple versions. The actual calculation of the score is done by a software implementation, colloquially called "tool" (e.g. "We developed a tool to assess genome-wide variant effects").

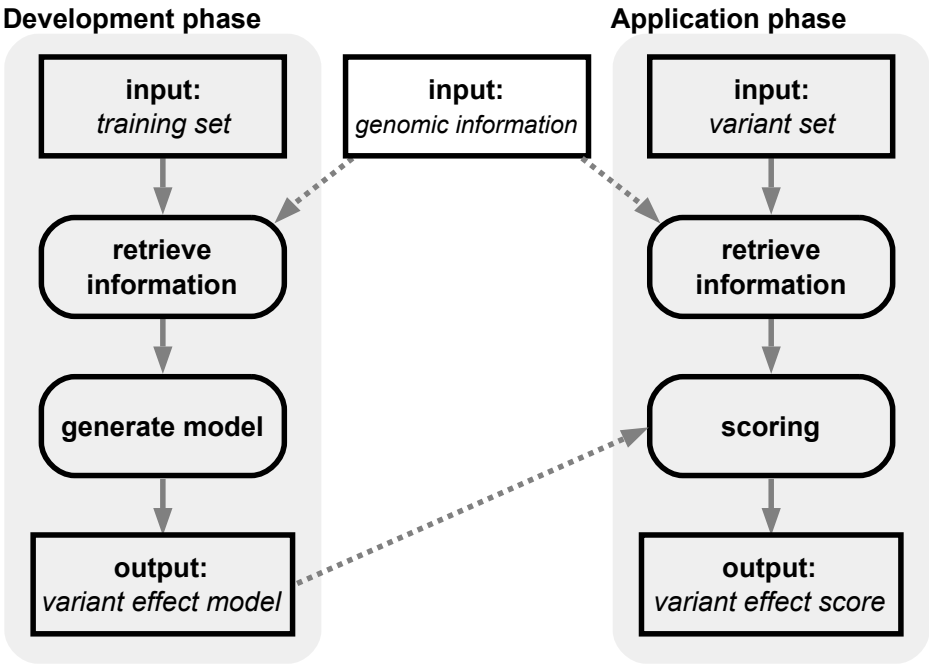

Figure 13: **Variant effect model and score.** The variant effect model is developed to predict a variant effect based on genomic information about genetic variants. The variant effect score is the metric generated by applying the variant effect model to any genomic variant.

Predicting variant effects

Each variant effect score has several defining properties: the type of effect predicted, the metric that is used to represent the effect, the types of information that are used

to perform the prediction, and the structure of underlying algorithms and general parameters of the model that performs the score calculation. Predicted effects can be the likelihood of an impact on the phenotype of the entire organism like pathogenicity or deleteriousness of the variant, or a single cellular process like the binding of a molecule to the DNA, RNA splicing, or enzyme kinetics. All these effects have in common that they are impossible to predict with simple deterministic methods, like the consequence of amino acid exchange on the protein sequence. The effect, or a matching metric, is studied in detail through a variant assay or animal model by measuring the consequence of every single variant from a given set. The measured effects can be used to develop the score and benchmark its performance. While raw or transformed measurements may be used for these tasks, many applications are based on a division into result classes: variants with and without an effect, sometimes further graded into severity levels. The predicted variant effect score is then optimized for distinguishing between these classes.

There are two fundamental concepts, based on which information the variant effects are predicted: based on either nucleotide sequence alone or integrating various genomic annotations. Sequence-based variant effect scores generally subtract the prediction for the alternative sequence of the variant from that of the reference sequence. Initially, many sequence-based variant effect scores were k-mer scores that assign one value to each possible DNA nucleotide combination of length k, with typical values of k ranging between 5 and 12 (Yeo & Burge, 2004; Lee *et al.*, 2015; Rosenberg *et al.*, 2015; di Iulio *et al.*, 2018). Advances in deep learning technologies have led to the prediction of effects from longer sequences, with recent neural networks integrating sequences that are thousands of nucleotides long (Jaganathan *et al.*, 2019; Avsec *et al.*, 2021). Many sequence-based variant effect predictors are specialized to a single biological processes like transcription factor binding (Avsec *et al.*, 2021), DNA splicing (Yeo & Burge, 2004; Rosenberg *et al.*, 2015; Jaganathan *et al.*, 2019; Cheng *et al.*, 2019), nucleosome positioning, ribosome loading (Sample *et al.*, 2019), or choice of poly-adenylation sites (Bogard *et al.*, 2019). More general application include missense (Sundaram *et al.*, 2018) and non-coding effect prediction (Zhou & Troyanskaya, 2015), including some like the k-mer score CDTS (di Iulio *et al.*, 2018) that are designed for genome-wide effect prediction.

In annotation-based variant effect scoring, the variant effect is predicted based on various types of genomic information (Fig. 15A). That information, referred to as annotations, is specific for the variant itself or has been generated for the genomic position of the variant. Annotations can be specific for one type of variant consequence (compare

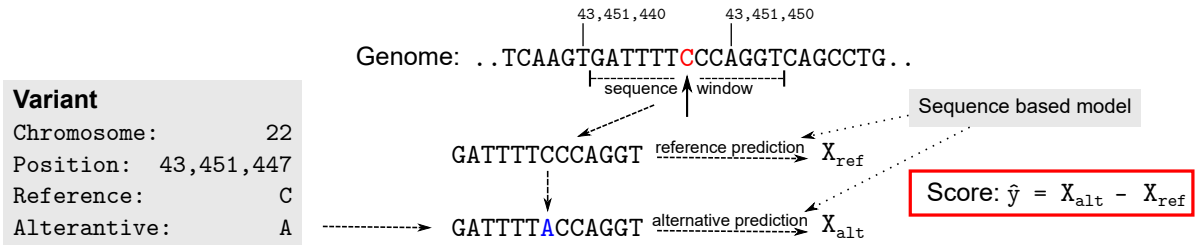

Figure 14: **Sequence-based variant effect model.** Entirely based on DNA sequence, a window around the variant is looked up in the genome. The alternative sequence is constructed from this reference sequence by exchanging the reference by the alternative nucleotides. The variant effect score is the difference of the sequence model prediction on both sequences.

Fig. 12) or generalized to all genomic variants. Sources of annotations are almost any kind of information that may be known or generated about the variant or its location. These include genomic sequencing readouts, properties of the DNA or protein sequence, and aggregated knowledge of genes and protein domains. As such, many annotations are also a type of variant effect that may be generated by another effect predictor. Meta classifiers integrate specialized process predictions and other variant effect scores as annotations into their prediction (Dong *et al.*, 2015; Ioannidis *et al.*, 2016). In total, a great number of annotations have been used in different variant effect scores, with every score being based on a distinct set. More detail about different annotations is given in chapter 3.3.

To generate the score, all annotations are retrieved, processed, and summarized by the model into the score. One way of summarizing many annotations into a single score is the linear model (Fig. 15C). All annotations are transformed into numeric values (Fig. 15B) and each is multiplied with a model parameter, the sum of which is the variant effect score. More complex model structures like binary trees (Fig. 15D) or deep neural networks enable the integration of feature interactions, which otherwise have to be created manually.

There is a continuous overlap between sequence and annotation-based variant effect scoring. For example, sequence-based scores may integrate gene annotations to derive the evaluated sequence or limit scoring to suitable genomic regions. Similarly, sequence properties like GC content, the observation of selected DNA sequence motifs, or entire, previously generated sequence-based scores can be annotations in annotation-based scores. Hard to classify are models like PolyPhen (Ramensky *et al.*, 2002) and SIFT (Ng & Henikoff, 2003), that are based on comparative sequence analysis, and share properties

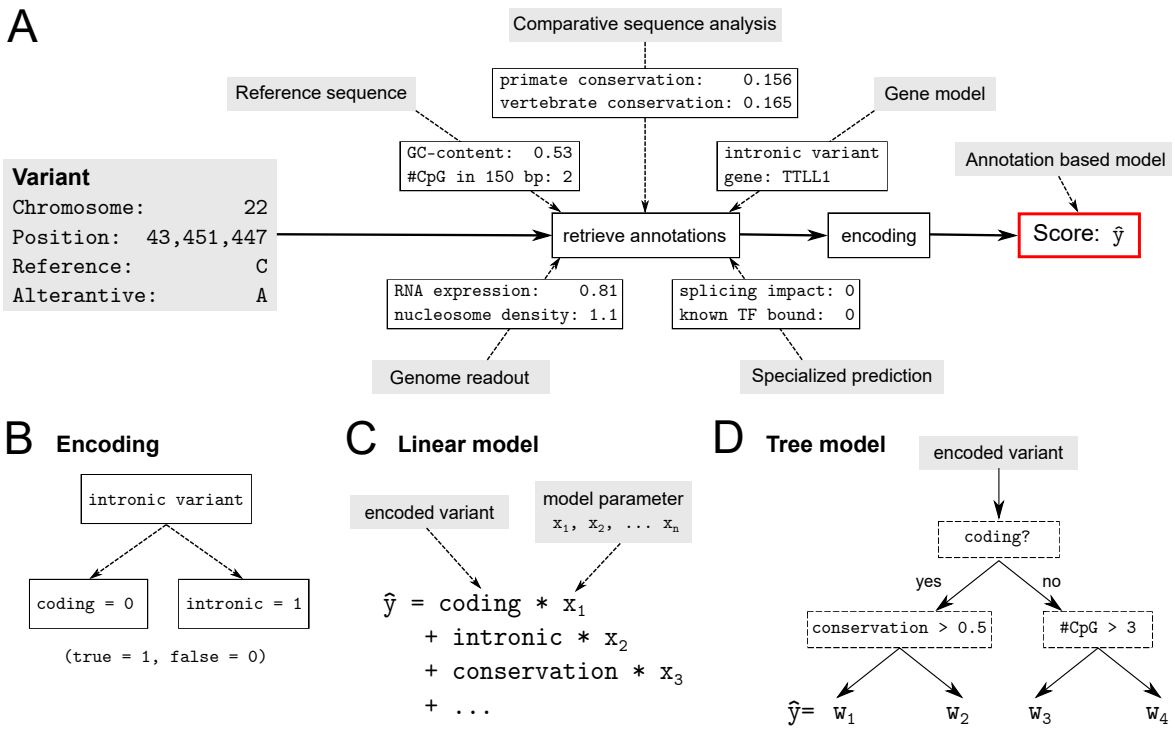

Figure 15: **Annotation-based variant effect model.** **A** based on variant position and substitution, different annotations are retrieved, encoded in a numeric vector and scored with the annotation-based model. **B** Encoding translates categorical annotations into numeric variables. The actual model can have different structures like a linear (**C**) or binary tree model (**D**).

from both types of variant effect scoring.

Learning to predict an effect

In principle, it would be possible to manually create a variant effect model based on expert knowledge. For many annotations, it has been well described how the encoded process translates to a phenotypic alteration. However, this is not the case for all annotations, even less so for sequence-based predictions. The assignment of variant information to effect is instead algorithmically optimized, or trained, using machine learning.

By the number of published scores, the most popular way of training such a machine learning model is based on labeled variants from various variant databases. These databases are easily accessible, genome-wide, and contain thousands of previously studied genomic variants. The model is trained by a machine learning algorithm that maximizes the ability to predict a previously determined metric or to distinguish between, in most

cases, two classes of variants: pathogenic and neutral.

Besides the annotations and the variant sets used for training, any variant effect score is defined by the algorithm that is used to generate the model. For this task, most variant effect models are using supervised machine learning based on labeled variants. Possible algorithms are interdependent with the final model structure. Binary tree based models are mostly trained via either random forests (Li *et al.*, 2009; Carter *et al.*, 2013; Ritchie *et al.*, 2014; Zhang *et al.*, 2015; Smedley *et al.*, 2016; Douville *et al.*, 2016; Ioannidis *et al.*, 2016) or gradient boosting trees (Jagadeesh *et al.*, 2016; Capriotti & Fariselli, 2017; Gray *et al.*, 2018), though other decision tree algorithms have been used (Hu & Ng, 2013; Jian *et al.*, 2014a). Linear models are trained with linear SVMs (Zhao *et al.*, 2013; Kircher *et al.*, 2014; Folkman *et al.*, 2015; Livingstone *et al.*, 2017) or logistic regression (Dong *et al.*, 2015; Huang *et al.*, 2017). More complex structures are possible with deep neural networks (Quang *et al.*, 2015; Xiong *et al.*, 2015). Other methods that are not based on supervised machine learning are using likelihood-ratio tests (Chun & Fay, 2009; Yandell *et al.*, 2011; Choi *et al.*, 2012; Fu *et al.*, 2014), Bayesian classifiers (Adzhubei *et al.*, 2010; Zhang *et al.*, 2014; Schwarz *et al.*, 2014), hidden Markov models (Shihab *et al.*, 2014) or clustering (Ionita-Laza *et al.*, 2016).

Using effect scores to diagnose patients

Variant effect scores are an important component in rare disease diagnosis pipelines (Zhu *et al.*, 2015). The score metrics are used to rank variants based on the likelihood to cause large phenotypic effects. Unfortunately, there seem to be no studies that quantify the impact of variant effect scores independent of linked variant analysis steps like filtering for common variants and known disease genes. Recent studies suggest that between 30% and 50% of novel genetic disease cases can be solved through WES followed by variant analysis (Farwell *et al.*, 2015; Stark *et al.*, 2016). This rate of diagnosis drops however for patients that could not be diagnosed in previous analyses (Boycott *et al.*, 2017), with follow-up WGS studies finding candidate variants in approximately one in four cases (Wright *et al.*, 2015). Using trio sequencing has led to increased diagnosis rates than sequencing of only the patient's genome (Farwell *et al.*, 2015).

3.2 Databases of analyzed variants

Prior knowledge plays an important part in the evaluation of a set of genetic variants. Nowadays, there are variant databases such as ClinVar (Landrum *et al.*, 2018), HGMD

(Stenson *et al.*, 2020) or OMIM (Amberger *et al.*, 2019) that contain tens of thousand of variants. The variants are classified by an ontology that includes an assessment of whether the variant may cause disease. The American College of Medical Genetics has defined guidelines for the assessment and assigns variants based on evidence in one of five different interpretation categories (from pathogenic to benign) (Richards *et al.*, 2015). In cases where a variant cannot be classified with absolute certainty due to missing evidence or incoherent data, it may be classified in the category 'uncertain significance' (Rehm *et al.*, 2015). For most variants, the database record includes an explanation for why a particular category was assigned, as well as additional information like population allele frequency, associated phenotypes, or medical case studies.

Genetic variant databases are steadily growing, with comprehensive databases like ClinVar as of 2021 containing in the order of 100,000 genetic variants while variant archives like dbSNP (Sayers *et al.*, 2011) or gnomAD have hundreds of million variants. In any case, all databases are far smaller than the 9 billion possible SNVs in the human genome, not to think about InDels and SVs where an ultimately unlimited number of possible combinations exists. All those variants are not found in databases or where evidence is inconclusive remain to be studied by genetic assays, phenotype association, and variant effect scoring.

Databases as a source for model training

Based on the assumption that already studied variants are a random subset of all genomic variants, databases are an important resource of labeled data when developing variant effect models. Especially "pathogenic" variants from ClinVar and HGMD have been used as positive training instances for developing variant effect models. Those have been set in contrast to variants classified as "benign" in the same database (Capriotti & Fariselli, 2017).

However, many regions of the genome have not yet been studied in depth. Hence, it is currently unknown to which extend variants in any database are subject to selection bias. Assuming that variant classification requires a prior study, especially the "benign" class may contain a large fraction of variants that were investigated for being causal of a disease. While any disease implications may have been cleared eventually and the variants are classified correctly as "benign", the selection, possibly based on some genomic feature, may have led to a different distribution than in the genomic background (Cooper *et al.*, 2010; Nykamp *et al.*, 2017). To avoid such biases, some models are

trained using common variants that are shared among many individuals or rare variants from healthy humans as a negative variant set. With that, they are following guideline recommendations that those variants should be regarded as neutral (MacArthur *et al.*, 2014; Richards *et al.*, 2015; Nykamp *et al.*, 2017).

In the same way that variants classified as "benign" are not representative of all neutral variants, variants classified as "pathogenic" went through specific ascertainment and most likely do not represent the true distribution of all disease-causing variants. Unbiased alternatives are data sets created from evolutionary changes between species genomes (Adzhubei *et al.*, 2010; Gulko *et al.*, 2015; Kircher *et al.*, 2014; Huang *et al.*, 2017; Sundaram *et al.*, 2018) or population variants (Ionita-Laza *et al.*, 2016; di Iulio *et al.*, 2018). Not all variants in such a set will be assigned the correct labels, which is why they may be termed proxy-labels. Possibly due to the larger absolute size of the data set, variant effect scores trained on proxy-labeled variants nevertheless perform reasonably well, as shown by the benchmarks published in the respective studies mentioned above.

3.3 Genomic annotations

Over the last decades, a lot of knowledge has been collected about different processes that are taking place in human cells. Long before the establishment of the human genome sequence, much of this knowledge had already been attributed to certain DNA sequences. As a result of the decoding of the human reference genome, these sequence-information relationships are now linked to defined coordinates of the genome and referred to as genome annotations (Stein, 2001). Annotation of a genomic variant is hence the process of retrieving linked data given the position of the variant in the genome.

Formats, access, and types

Some annotations can be derived from other annotations, like the amino acid exchange caused by a genomic variant that depends on the annotation of a gene, the reference amino acid, and the substitution taking place. All other annotations that are not derived are generally stored in annotation data files. These files consist of individual data points that each are related to one position, or coordinate, in the genome. The genome position can be a single bp or a continuous stretch of DNA, from a start position to an end position (smaller to larger coordinate). For annotations where the direction is important, an optional strand information encodes whether an annotation like a gene refers to the DNA strand that is encoded in the genome reference sequence (+) or the respective reverse

complement (−). Similarly, a phasing parameter can specify when multiple elements are located on the same haplotype. In addition to that, positions can be specific to a single DNA substitution in which reference and the substituting alternative sequence are listed. The information that is encoded by the annotation may be of numeric value (e.g. expression is 4.5), categorical (e.g. segment is a regulatory element), or string-based (e.g. gene name is YAP1). In some cases, the position alone is sufficient as it encodes the existence of the annotation at that position. Multiple information sources may be summarized based on shared positions in a single annotation file.

Although standards for biological information storage are not a recent development (Ashburner *et al.*, 2000), raw annotation data is stored in files of almost any format. Manuscript data is often published in a way that is practical to the authors, for example in a Microsoft Excel sheet or an unspecified text file. Nevertheless, a number of standardized formats are popular and used for many large annotation data set. The Browser Extensible Data (BED, `https://genome.ucsc.edu/FAQ/FAQformat.html#format1`) format is designed for annotating genomic range data. In addition to three required position fields (chromosome, start, and end), a BED file contains up to 9 to specify information like element name or strand information. If only a score value is assigned to each range, the simpler bedGraph format (`https://genome.ucsc.edu/goldenPath/help/bedgraph.html`) limits columns to the three for the position in addition to a fourth for the score itself. Other formats to store genomic range information are the General Feature Format (GFF) and its derivatives Gene Transfer Format (GTF) and Genome Variation Format (GVF, Reese *et al.*, 2010). All three consist of nine columns that encode a genomic position and the annotated features name, source plus additional attributes (`https://genome.ucsc.edu/FAQ/FAQformat.html#format3`). While GFF has been developed to annotate any type of genome element, GTF specifically addresses data where multiple attributes are stored for each element. GVF is optimized to describe sequence alterations in association with a stored attribute. Continuous data with constant element size is commonly presented in wiggle format (WIG, `https://genome.ucsc.edu/FAQ/FAQformat.html#format6`). WIG has the advantage that position information of subsequent values can be omitted, reducing file size and increasing access speed. The Variant Call Format (VCF, Danecek *et al.*, 2011) stores genetic variants in coherence with information about these variants and is especially flexible regarding the annotation of multiple different annotations in a single file. All described data formats are stored in tab-delimited ASCII text files and therefore read-

able and writable with any text editor. The size of the raw text files is reduced via compression, generally with gzip (`https://gnu.org/software/gzip`).

Loading large annotation files is time-consuming and often limited by the space of available working memory. Efficient collection of annotations for a set of variants is hence constrained through the necessary files accesses. In order to avoid loading the entire files and reading until the required position, a file index for each file enables direct file access so that data is read only at the requested position. These file indices are binary-tree derived data structures that contain file pointers to defined genomic intervals. Tabix (Li, 2011) is a specialized bioinformatics software to generate these indices for compressed files in formats like BED, GFF, and VCF. Alternatives are dedicated file formats such as bigWig and bigBed that include an index at the beginning of the data file (Kent *et al.*, 2010). Functionally identical are indices from relational databases such as SQL. Specialized variant annotation software like Annovar (Wang *et al.*, 2010) or Ensembl VEP (McLaren *et al.*, 2016) employ such a database to efficiently annotate variants.

Sources of annotations

Any annotation-based variant evaluation depends on many different annotations. Some annotations are measured directly as raw biological readout while others are derived by integrating various sets of raw data. Common to all is that they represent some type of cellular process or genomic feature.

The first considerations when annotating a genomic variant are other genomic elements that have been located at the same position. These elements provide a general inference of what kind of consequence (see also Fig. 12) the variant may have and thus what kind of other annotations may be relevant. The location of all transcripts that are transcribed from a genome, associated exons and introns, and how these fit into a limited set of genes is called a gene annotation. Human gene annotation builds are published in databases like GENCODE (Harrow *et al.*, 2012) and RefSeq (Pruitt, 2004). Information like gene names, transcript identifiers, and encoded proteins are referenced in the gene annotation and can be used to obtain further annotations.

There are plenty of genomic features that may be annotated to variants in protein-coding regions. Given the annotation of the transcribed sequence, sequence effects are simple to predict. From these, missense effects may be analyzed for changes in biochemical and biophysical properties of the involved amino acids. The same substitution may have different effects at different positions in the protein, so that the function of

the associated protein domain or secondary structures like alpha helices and beta sheets may be considered. Databases like PDB, STRING and UniProt are a source of three-dimensional structures (wwPDB consortium, 2019), protein-protein interactions (von Mering *et al.*, 2005) or known post-translational modifications (Khoury *et al.*, 2011). Homologous proteins and domains of known function may enable conclusions that can be adapted to the variant of interest (Loewenstein *et al.*, 2009). While there is recent progress in predicting protein structure directly from sequence (Jumper *et al.*, 2021), it is currently unclear how predictive these structures are for functional changes.

In the absence of coding sequence effects, other annotations are emphasized when synonymous or intronic variants are evaluated. These variants impact gene expression via codon usage bias (Liu *et al.*, 2021) and RNA splicing, the assembly of multiple exons into a mRNA. Alternative splicing of a single gene into multiple isoforms is one major source of the complexity in the human proteome (Ule & Blencowe, 2019). Many components contribute to "correct" RNA splicing, including donor and acceptor splice sites, branch points, and binding sites for small nuclear ribonucleoprotein particles as well as the 3D conformation of the RNA, which may be impacted by genomic variants (Lee & Rio, 2015).

Annotations that link gene expression via changes in the mRNA levels may be considered for variants in untranslated regions (UTRs). The 5' UTR of a transcript can be affected by upstream open reading frames (Whiffin *et al.*, 2020). Ribosomal loading is affected by the Kozak sequence (Sample *et al.*, 2019). Binding sites for miRNAs and RNA binding proteins have been found important annotations when evaluating variants that are located in the 3' UTR (Griesemer *et al.*, 2021). However, the by far largest expression changes in 3' UTRs seem to be caused by variants that alter existing polyadenylation sites (Bogard *et al.*, 2019), encouraging the annotation of such elements.

Compared to transcribed regions, it has taken a long time to develop comprehensive, genome-wide annotations of non-coding variants. The impact of transcription factor (TF) binding on gene regulation has been known for a long time (Mitchell & Tjian, 1989), and TF motif databases like JASPAR (Sandelin *et al.*, 2004) and DNA shape predictions (Zhou *et al.*, 2013) annotate possible binding sites genome-wide. However, it is impossible to separate functionally active sites based on DNA sequence motifs alone, even though these are certainly significantly enriched (Elkon, 2003; Wang *et al.*, 2018; Zeitlinger, 2020). Over the last 15 years, the encyclopedia of DNA elements (ENCODE) and similar projects have generated a large collection of genome readouts of transcrip-

tion factor binding, histone modification, and chromatin state. Integrative functional analyses based on these data have defined candidate cis-regulatory elements that can now be annotated from databases like SCREEN ENCODE Project Consortium *et al.*, 2020. The generated data serves also as input for segmentation and genome annotation algorithms like ChromHMM (Ernst & Kellis, 2012) and Segway (Hoffman *et al.*, 2012) that divide the entire genome into distinct functional classes. The characterization of chromatin structures like topological associated domains has greatly increased the understanding of regulatory element interaction (Hu & Tee, 2017), although the definitive association between a functional element and the regulated gene remains an open challenge (Schmidt *et al.*, 2020).

A central component in most annotation-based variant effect scores, independent of other genome features, is conservation: Conservation scores like phastCons (Siepel *et al.*, 2005), phyloP (Pollard *et al.*, 2010), GERP (Davydov *et al.*, 2010) and SiPhy (Garber *et al.*, 2009) measure the stability of genome segments over evolutionary time scales. Regions of the genome sequence from different species which align very well are considered highly conserved. In contrast, regions where there are no or few closely related species genomes align, or where there are frequent nucleotide mismatches in the aligned genomes, are of low conservation. While mutations happen by chance and may affect all positions of the genome, those in functional regions will lower the reproductive success of its carrier and are therefore selected against. Survivorship bias causes mutations to accumulate in regions that do not affect function (Havrilla *et al.*, 2019). Consequently, conservation scores are a stochastic measure of mutational tolerance and identify positions in a DNA sequence that are absolutely necessary for the function of the respective genome segment (Lindblad-Toh *et al.*, 2011; Finucane *et al.*, 2015).

In summary, there is a large number of annotations that may be considered for variant effect scoring. Those pointed out here serve only as examples for a much larger variety, with constant publishing of new annotations. Variant effect scores that predict cellular processes may be themselves the source of an annotation in other, more general scores. Arguments for or against single annotations are dependent on the individual variant effect score. Considerations should be mostly regarding the association with the predicted variant effect, correlation with other annotations, and how far the annotation may be biased for or limited to previously studied genes. It is, therefore, no surprise that every variant effect score will use a slightly different set of annotations.

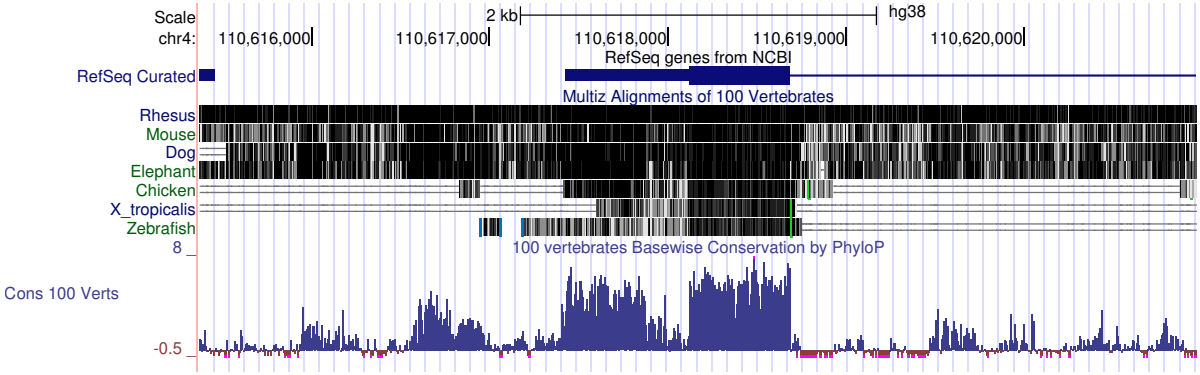

Figure 16: **Genome conservation in UCSC genome browser.** Screenshot showing the transcription start site of the gene PITX2. The center of the plot are alignments of other vertebrate genomes to the human reference. In the UTR and coding region of PITX2, more species genomes like chicken and zebrafish align than in non-coding regions. More and less variable alignments lead to higher conservation, here represented exemplary by phyloP 100 vertebrate conservation.

3.4 Variant score implementations

As of 2021, an overwhelming number of computational variant effect scores, associated software, and web server have been published. Every score has been developed with a certain focus like coding or non-coding regions of the genome, certain effect categories, variants of unknown significance, or InDels and SVs. Parameters like spatial resolution are influenced by the used annotations, while architecture optimizations enable novel insights based on previously known data.

Coding variant scores

Coding variants that result in changes to the amino acid sequence of a protein and are a primary cause of large variant effects (Cargill *et al.*, 1999; Sunyaev *et al.*, 2001). A large fraction among coding variants are missense variants, in place substitutions of amino acids in protein-coding genes. While old and new protein sequences are simple to predict based on nucleotide triplets, the coding effect score predicts pathogenicity that may be caused by the altered protein.

The first considerations regarding coding variants were amino acid substitution matrices like Grantham (Grantham, 1974), PAM (Dayhoff *et al.*, 1979) and BLOSUM (Henikoff & Henikoff, 1992) that are based on evolutionary substitution likelihoods. With the rise of comparative genome alignments, these were combined with genome conservation

in methods like PolyPhen (Ramensky *et al.*, 2002) and SIFT (Ng & Henikoff, 2003). Recent coding variant scores integrate knowledge about biophysical and biochemical characteristics of the altered amino acids, post-translational modifications, protein domains, protein classes, homologous proteins, and enzymatic assays into the prediction (Adzhubei *et al.*, 2010; Choi *et al.*, 2012; Carter *et al.*, 2013; Gray *et al.*, 2018).

Protein truncating variants (PTVs), coding variants that cause a novel stop codon, the loss of a start codon, a frame-shift or change in splicing, have on average equal or more constraint than missense variants (Exome Aggregation Consortium *et al.*, 2016). While a PTV close to the reference stop codon may have less of an effect than one closer to the start of the gene (Nagy & Maquat, 1998), the assessment of PTVs is less variant-specific since the partial or full loss of protein function can be assumed in almost all cases (Rivas *et al.*, 2015). The main consideration regarding PTVs are hence properties of the encoded gene like essentiality and haploinsufficiency (Cassa *et al.*, 2017; Rausell *et al.*, 2020).

Of limited concern regarding protein function are synonymous variants that would rather influence gene regulation than alter protein function (Hunt *et al.*, 2014; Exome Aggregation Consortium *et al.*, 2016). Considering their impact on RNA splicing, many coding variant effect scores nevertheless integrate annotations to predict synonymous variant effects. Variants that cause the loss of a stop codon are often neglected for being statistically rare (Hamby *et al.*, 2011) and generally evaluated like missense variants without dedicated annotations (e.g. Adzhubei *et al.* (2010); Kircher *et al.* (2014); Ionita-Laza *et al.* (2016); Gray *et al.* (2018).

Specialized scores and non-coding variants

Many variant scores are predicting effects on a single molecular process like RNA splicing (Jian *et al.*, 2014b; Cheung *et al.*, 2019) or transcription factor binding (Alipanahi *et al.*, 2015). The generated scores are not replacements of pathogenicity but rather the effect on the process independent of organism-wide impact. Variant sets that are used to train specialized models are defined in molecular assays that sample the modeled effect like Deep Mutational Scanning (Gray *et al.*, 2018) and STARR-seq (Muerdter *et al.*, 2015). Another alternative are associated labels based on genome-wide readouts like peaks in ChIP and DNase-seq data set (Zhou & Troyanskaya, 2015; Kelley *et al.*, 2016).

Findings from GWAS analyses indicate that a majority of phenotype-associated variants may be located in non-coding regions of the genome (Edwards *et al.*, 2013). How-

ever, the number of possible non-coding variants is much bigger than that of coding variants, increasing the number of candidate pathogenic non-coding variants. As a consequence, a comparatively small number of non-coding variants have been found causal of genetic disease and added to databases such as ClinVar (Makrythanasis & Antonarakis, 2013; Smedley *et al.*, 2016).

In addition to that, many annotations that are key for coding prediction are unsuitable for non-coding parts of the genome (Drubay *et al.*, 2018). Many regulatory regions of the genome are recently evolved (Ponting, 2017), so that genomic conservation, a central component of coding effect prediction, performs poorly on variants in regulatory elements (Kircher *et al.*, 2019). Due to the absence of good alternatives, many (Lee *et al.*, 2015; Zhou & Troyanskaya, 2015; Kelley *et al.*, 2016; Avsec *et al.*, 2021), but not all (Smedley *et al.*, 2016; Huang *et al.*, 2017), non-coding effect scores are sequence-based.

Genome-wide variant effect scoring

Genome-wide models combine the evaluation of variants in coding and non-coding regions of the genome in a single metric. Due to the limited number of non-coding variants in databases like ClinVar or HGMD, they are generally trained on other training set such as variants detected in populations sequencing (Ionita-Laza *et al.*, 2016) or derived by comparison to other genomes (Kircher *et al.*, 2014). The attribute genome-wide further implies that annotations have to be considered especially carefully in order not to introduce observation or selection biases. Due to the number of covered molecular consequences, a genome-wide pathogenicity effect score can depend on hundreds of separate annotations.

Limits to "genome-wide"

Unfortunately, genome-wide does not mean that all variants in the human genome can be scored or will be scored equally. Besides regions of the genome that are missing from current reference genomes, the mapping of sequencing reads has higher error rates in some regions of the defined genomic sequence than in others. Reference genomes like GRCh38 further include alternative contigs, parts of the genome where multiple different haplotypes coexist at the same genomic location. Alternative contigs may lead to ambiguity in read or sequence alignments that may influence the generation of many genomic annotations. Similar restrictions apply to the human gonosomes X and Y. Sex-dependent coverage and different evolutionary selection (e.g limited recombination in

chromosome Y) alter the profile of many annotations which impacts the resulting local score distribution.

One part of the human reference genome where it has been recommended that variants are analyzed separately from the rest of the genome is the mitochondrion (McCormick *et al.*, 2020). The reasons are that the mitochondrial DNA is short in length compared to the rest of the genome, has a high density of housekeeping genes, and is only inherited from the mother. While mitochondrial variants can be scored with genome-wide scores, variant effect scoring specialized to the mitochondrial genome has been shown to lead to better results (Castellana *et al.*, 2017).

InDel and SV

While pathogenicity of all structural variant types is generally accepted, most variant effect scores are limited to SNVs. Responsible for this focus may be problems in variant calling (Chen *et al.*, 2019) that causes a biased representation of all none-SNV types in databases from which variants are selected for model training. Another problem is the proper assignment of many annotations variants that are affecting more than one nucleotide and cause changes in the sequence length. Finally, the limited number of classified variants is also a bottleneck regarding score assessment.

Nevertheless, several methods have been developed for variant effect prediction on small InDels (generally defined as > 50 bp). Some methods are able to score different types of insertion or deletion events (Zia & Moses, 2011), though often restricted to coding regions of the genome (Douville *et al.*, 2016). Others are specialized on frameshift (Zhang *et al.*, 2014) or non-frameshift InDels (Bermejo-Das-Neves *et al.*, 2014), or were released in separate models for different types (Hu & Ng, 2012, 2013; Zhao *et al.*, 2013; Folkman *et al.*, 2015).

In addition to dedicated InDel scores, some methods score SNVs and InDels using one general model (Adzhubei *et al.*, 2010; Choi *et al.*, 2012; Hu *et al.*, 2013; Kircher *et al.*, 2014; Schwarz *et al.*, 2014). While there are further overlaps between SNV and InDel scores that were created by the same group of authors (e.g Carter *et al.*, 2013 and Douville *et al.*, 2016) that often share much of the underlying method, no joined score created by only a single model is implemented in those cases. Notably, using a single scale for SNVs and InDels is rarely validated.

For SVs, only a number of specialized scores have been published (Ganel *et al.*, 2016; Geoffroy *et al.*, 2018; Huynh & Hormozdiari, 2019; Kumar *et al.*, 2020).

Variant effect prediction in other species

Most annotations and many assays have been generated only for the human genome, and are often limited to the coordinates of the genome build GRCh37. In concordance, almost all variant effect scores are either generic or established only for human variants (Wagih *et al.*, 2018). This focus is simple to explain as most interest in molecular principles is coming from a perspective of human medical use. Even in relatively well-studied model organisms, most interest is in genome segments that are homologous to the human sequence, regions where variant effect scores can be transferred with genomic liftover (Liu *et al.*, 2016). Species-specific variant effect scores have been developed for mouse (Groß *et al.*, 2018) and dog (Capriotti *et al.*, 2019) and show better performance than lifted scores. Agricultural interest has further lead to the development of models for important livestock species (Groß *et al.*, 2020a,b).

Comparisons between scores

Independent of the metrics used (Vihinen, 2012), comparing different variant score models is complicated. Every score has a slightly different set of annotations or variants used for training, so that score differences cannot be attributed to a single source. When published, almost every score is described as having the best performance on presented benchmarks or at minimum as being an advantage in previously missed variant categories. In the shown benchmarks, it happens that other tested scores are using parts of the tested variant set as input to model training. An appealing solution to training-test overlaps are multiple models, one for each chromosome, so that all genomic variants can be scored as hold-out from the training set (Smedley *et al.*, 2016). However, this leave-one-chromosome-out strategy has the limitation that multiple models correspond to a single score and is rarely applied.

In response to missing comparability, general benchmark sets have been proposed that should not be used for model training (`https://github.com/quinlan-lab/pathoscore`, Sundaram *et al.*, 2018). Independent of how practical such an idea is, once a benchmark set is well-known, every model developer may, consciously or unconsciously, optimize their novel model for that data set. An always temporary solution to such a performance race are novel, independent, and not yet fully released benchmark sets like those presented as CAGI challenges (`https://genomeinterpretation.org`).

3.5 CADD

The word CADD, without further specification, generally addresses both the CADD model, a machine learning classifier to evaluate genomic variants, and the CADD score, the corresponding variant effect score. The acronym CADD is short for "Combined Annotation - Dependent Depletion". Both the model and the score were originally published by Kircher *et al.* (2014) in Nature genetics and via the web server `https://cadd.gs.washington.edu`. CADD is able to predict the pathogenicity of SNVs and short InDels genome-wide.

Every variant effect score has some specialty to their machine learning or how annotations are selected and processed. Crucial for CADD are two components: a huge training data set that includes more than 30 million genomic SNVs and InDels, and more than 100 different annotations. The main framework is outlined in Figure 17.

Derived variants

The training set for most other variant classifiers is selected from variant databases like ClinVar or HGMD. As the human genome is too large to be comprehensively studied, variants that are easier to find or located in genes that have been found interesting in the past are enriched in those databases. Any variant classifier developed based on such a data set integrates this ascertainment bias. In contrast, variants used for the training of CADD are not taken from curated data. Instead, SNVs and InDels are chosen from the human genome based on the principle of purifying selection.

A random de novo mutation that appears in an individual can affect genome function and with that the individual's genetic fitness in two ways, either beneficial or adverse. Variants with a beneficial effect will statistically increase the reproductive success of the individual, be inherited by their progeny, spread through the population, and eventually become fixed in all individuals. A variant with an adverse or deleterious effect will decrease reproductive success and, sooner or later, cause the variant to go extinct. This selection against deleterious variants and for beneficial variants is termed purifying selection. The genome sequence of any biological organism has been under natural selection for billions of years. Due to gradual adaption to the current environment, almost all novel variants that affect genetic fitness will be deleterious as any beneficial variant has ultimately become fixed while deleterious variants can appear again and again. Variants that are observed in a population or between two or more species are depleted of deleterious variants even though there is variance in phenotype. The majority of

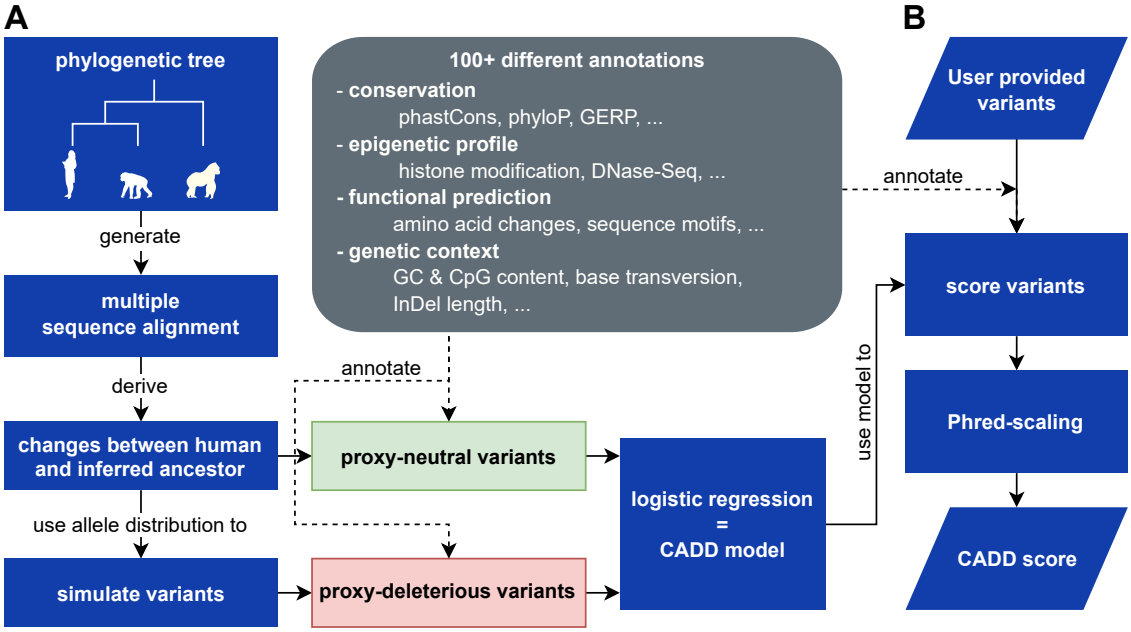

Figure 17: **Overview of the CADD framework.** (A) Training a CADD model relies on two sets of variants; a proxy-neutral set of human-derived variants and a proxy-deleterious set of randomly drawn, simulated variants that match the sequence composition of the human-derived set. Using more than 100 diverse annotations, a logistic regression classifier is trained to differentiate between proxy-neutral and proxy-deleterious variants. That classifier is the CADD model. (B) Variant sets are provided as VCF files. Variants are evaluated based on chromosome, chromosome position, reference allele, and alternative allele columns from these files. Scores are either retrieved from prescored files or variants are fully annotated and the CADD score is calculated by the model. The CADD Phred-score, the relative ranking of each variant compared to all SNVs in the genome, is looked up in a conversion table, and both scores are returned.

variants are thought to have no impact on genetic fitness. These neutral variants do not affect reproductive success and increase or decrease in frequency by chance. Based on Bayesian statistics, it follows that most neutral variants will go extinct within a few generations while only very few variants will randomly spread through the entire population. Estimates have put the total fraction of the human genome that is under strong purifying selection in the range of five to ten percent (Lindblad-Toh *et al.*, 2011; Rands *et al.*, 2014). It follows that a sample of variants generated by applying random changes to the genome sequence will contain a corresponding fraction of variants that are under

purifying selection.

The CADD training set is derived based on these assumptions. The positive class, called proxy-neutral or human-derived, consists of differences between the human genome and that of an inferred ancestor. Based on Ensembl EPO alignments (Herrero *et al.*, 2016), a genome sequence of the common ancestor of humans and the great apes is inferred. The proxy-neutral set is obtained by selecting all variants that are absent in the ancestral genome but have an allele frequency of greater than 95% in the human population. The negative, or proxy-deleterious, class is sampled from the genome under the following constraints: In order to prevent the final model to classify variants based on properties like allele bias or chromosomal location, SNV allele substitutions frequencies, number of SNVs per 100 kb of the genome, and number of CpGs affected are modeled after the numbers from the proxy-neutral class. InDels are matched by length, as well as by the number of variants per 100 kb of the genome. The labels "proxy-neutral" and "proxy-deleterious" are used for the two sets as both are highly heterogeneous and are only enriched or depleted of deleterious variants respectively (see chapter 6.1 for a mislabeling estimate). In the case of the human genome build GRCh37, each class consists of 15 million SNVs and 1.8 million InDels.

Annotations

All 33 million variants are annotated with more than 100 different annotations. Annotations that are used in CADD come from a wide range of biological processes and assays, and are represented by diverse data types. Many make a prediction about a single process, such as DNA splicing, and are available only for a subset of variants. Major sources of annotations include the Ensembl Variant Effect Predictor (VEP, McLaren *et al.*, 2016), and data portals like the UCSC genome browser (Casper *et al.*, 2018), ENCODE (ENCODE Project Consortium, 2012; Davis *et al.*, 2018) and gnomAD (Karczewski *et al.*, 2020). Examples for annotations from these data collections are transcript information like distance to the closest exon-intron boundary, transcription factor binding or expression levels in commonly studied cell lines, and amino acid substitution scores for protein-coding sequences like SIFT (Ng & Henikoff, 2003) and PolyPhen-2 (Adzhubei *et al.*, 2010). Other annotations are downloaded directly from project specific repositories including Segway (Hoffman *et al.*, 2012), GERP (Davydov *et al.*, 2010) and microRNA binding predictions from mirSVR (Betel *et al.*, 2010) and targetScan (Lewis *et al.*, 2005; Agarwal *et al.*, 2015). A few annotations are specially adapted

for their use in CADD, running the software provided by the original authors. Conservation scores like phastCons (Siepel *et al.*, 2005), phyloP (Pollard *et al.*, 2010) and GERP are computed from multiple sequence alignments similar to those used to find the proxy-neutral set of human-derived training variants. In order to prevent those annotations to be biased for variants from this training set, phyloP and phastCons have to be specifically calculated without taking the human reference sequence into account. The GERP conservation score would only be affected if more than one genome sequence was mutated in the multiple sequence alignment and therefore does not need correction. Lists of all annotations used in each CADD version are available from the release notes on the CADD web server.

Model and score

After annotation, the missing feature values are imputed and the entire data set is converted into one large training matrix. A single model is fitted and can then be applied to any genomic variant. The output of the model is the CADD raw score. The numeric range of the raw score is model-specific and does not have an absolute meaning. It only gets a relative meaning via the application of the model to all ~ 9 billion possible SNVs of the human reference genome. A Phred-score is calculated from the relative rank of each variant and is linked to the corresponding CADD raw scores via a conversion table.

History of CADD versions

Since its inception, the details of model training have changed multiple times. CADD v1.0 was not trained as a single machine learning model. Instead, ten linear SVMs were trained using the machine learning library LIBOCAS (Franc & Sonnenburg, 2009), each using the same human-derived set but different sets of simulated variants for training. Models were trained for 2.000 iterations and a total run time of 70 hours but did not converge to a stable optimum. To offset the missing convergence, coefficients of the final model were calculated by averaging all ten models.

GraphLab (Low *et al.*, 2014) was adopted as machine learning library beginning with CADD version 1.1. The SVM used for training the model was replaced by a logistic regression classifier. Similar to version 1.0, ten classifiers were trained for eight training iterations and merged into a single model. CADD v1.1 further extended the initial set of features by five annotations to specifically improve the prediction of non-coding variants

Table 4: Novel annotations from CADD version 1.1

chromHMM	(Ernst & Kellis, 2012), functional genomic segmentation model based on 127 cell types from the ENCODE and Roadmap Epigenomics projects
DNA shape factors	(Zhou *et al.*, 2013), predicting the DNA secondary structure of sequence heptamers
miRNA binding sites	predictions from mirSVR (Betel *et al.*, 2010) and targetScanS (Lewis *et al.*, 2005; Agarwal *et al.*, 2015)
Mutation index	(Michaelson *et al.*, 2012), index of genome-wide mutability
Protein domain annotation	provided by Ensembl VEP based on protein databases such as PANTHER (Thomas *et al.*, 2003) and Pfam (Finn *et al.*, 2010)

(Tab. 4). One of these annotations, the DNA shape factors, was annotated incorrectly in the provided prescored SNV files. This mistake was corrected in the follow-up release CADD v1.2.

For the next release, the extraction of human-derived variants was based on new Ensembl EPO 6 primate alignments and extended to genomic segments with missing or more distantly related (Scally *et al.*, 2012) chimpanzee sequences by comparing the human sequence to those in the gorilla genome. It also fixed several issues with the InDel extraction. A single logistic regression model was trained from the new training set of 30 million SNVs and 3 million short InDels, terminating model training after 10 iterations and dropping the model averaging. The new model used the same annotations as the previous release and was released as CADD version 1.3.

Advantages and disadvantages

Due to the heterogeneous training data set that has not been selected and curated by any human but rather is generated by an algorithm directly from sequence alignment, the CADD model is not biased for certain variant categories. The size of that set and diversity of variants makes it possible to train a model from many annotations that correspond to many different variant properties and consequences, many of which may not be represented in a selected set. However, due to the set of human-derived variants

being defined as variants where the ancestral sequence mutated into the reference, i.e. reference and alternative sequence are switched, representation is limited in regards to some variant consequences like gain of a transcription start site. Some annotations further need to be corrected for recent evolutionary changes as those are part of the training data set.

It may be speculated that the large fraction of mislabeling in the training set may influence the adoption of some annotations. Contrary to that, mislabeling may reduce overfitting as, similar to drop-out, the wrongly classified variants serve as regularizing factors for very strong model features, though none of these effects have ever been shown. In addition to that, the CADD score has been criticized for not discriminating properly between variants used in training and variants appearing in common reference set (Heijl *et al.*, 2020), although enrichment of any of the two variant classes in a benchmark set is likely to be due to evolutionary constraint and not man-made selection.

Influence on other scores

As one of the most influential variant effect scores, other research groups not related to the authors have developed similar scores. DANN uses the annotated training data from CADD v1.0 in combination with a deep neural network (Quang *et al.*, 2015). CAPICE (Li *et al.*, 2020) has combined most of the CADD annotation set with a different training set in a gradient boosting classifier. The approach of using evolutionary-derived variants as training set of a binary classifier has been adopted for other model organism such as mouse (Groß *et al.*, 2018), pig (Groß *et al.*, 2020b) and chicken (Groß *et al.*, 2020a). In contrast to CADD which is defined for SNVs and InDels up to a length of 50 bp, all listed, derivative scores are limited to SNVs.

4 Considerations for the development of genome-wide variant effect scores

When I started on this project in 2017, the variant effect score CADD was already a great success. In the three years since its initial publication, four updates had been released, which included changes to the machine learning framework and the addition of new annotations. The score had attracted hundreds of users who used variant effect scoring via the web server and the script release. Altogether, this had led to more than a thousand citations of CADD in other scientific publications.

Unfortunately, the old CADD scripts up to version 1.3 had become difficult to manage. Internally, the entire project was best described as an organically grown collection of scripts and annotation files that were continuously executed through a well-documented but often complicated list of shell scripts. All annotations, dependencies, and model parameters were hard coded and had to be manually adjusted in the source code. Machine learning was dependent on the closed-source library GraphLab, which hid some crucial processing steps such as data set scaling. The biggest problem, however, were the many separate processing steps required to generate a new CADD release: Annotation, model training, benchmarking, model selection, and genome-wide scoring each consisted of separate steps that had to be manually executed and checked for completion one after the other. All of this led to a complicated setup process with manual downloading of scripts, annotation files, and dependencies, prompting one competitor to declare: "CADD is a beast!".

In order to continue working on CADD and improve variant effect scoring, we envisioned a new workflow. That workflow was to involve all the necessary steps to generate new CADD releases: From annotating variants, learning and optimizing the classifier, scoring variants, to finally making the generated variant effect score available to the scientific community. The framework that is executed by the workflow was imagined to follow the principles of reproducibility, extensibility, and simplicity. This chapter summarizes our considerations in designing such a workflow before the next chapter describes its implementation.

4.1 Principles for a variant effect scoring framework

In order to build an applicable variant scoring model, several steps have to be performed in succession: The training data set has to be selected, the annotations chosen, all this

information concentrated into a numeric matrix, which is then used to train a machine learning model. As models can be trained with different parameter values, these models are evaluated in terms of strengths and weaknesses until one final model is selected. That model is finally deployed, in order to enable the application of the developed variant effect score in practice. Although most of the involved steps require careful planning and evaluation, the entire process can be summarized into one workflow that sequentially executes the different steps. Continuous blocks are organized into software pipelines that are controlled by a central configuration. The results of each pipeline are carefully evaluated before the next steps of the workflow are started.

For a scientific project that consists of many different scripts and interacts with existing data, software, and software libraries, it seems obvious to follow good practices of software development. Good practices for software development concern the state of the software itself, such as how the code is written and documented, version control, how dependencies are managed, and how the various programs interact with other software and data, adopting Application Programming Interface (API) descriptions and predefined file formats. While all of that may seem to slow down the development of the source code during the initial writing process, it simplifies the management of the entire, growing codebase in the long run. Further, a well-managed source code is easier to replicate, so each pipeline can be ported to other computing systems and achieve the same results there. Replication is important for any scientific project so that other members of the scientific community can understand and reproduce the results (Peng, 2011). Reproducibility goes hand in hand with stability in that small changes in the analysis should produce comparable results, and reduce outlier results like the figurative butterfly effect, where small differences in the input lead to vastly different outcomes.

Compliance with standards is, obviously, only necessary for the acceptance of valid data. It does not necessarily mean that files that are not properly standardized have to be rejected. For example, a variant effect scoring script may accept as input a simple tab-delimited file with the required fields chromosome, position, reference, and alternative, even though it does not contain the complete header that a valid VCF file should have according to the file standard (Danecek *et al.*, 2011). While this may be disadvantageous for the detection of corrupted input files, it lowers the entry barrier for novel users who want to evaluate "just" a few variants.

Besides being flexible to extend and compatible with other applications and systems, a workflow for scoring variant effects should be fast. The annotation of each variant in the

data set is independent of all other variants and can therefore be executed in data-parallel processes. Equally, multiple models that test different features or model parameters can be trained and applied simultaneously. Pipeline frameworks like Snakemake are useful for managing parallelization and provide interfaces to common scheduling and distribution systems (Mölder *et al.*, 2021).

4.2 Annotations

Depending on the method, there can be hundreds of different annotations for every genomic variant. These annotations are derived from various data sources, including biological readouts, statistical analyses, and sequence predictions. How the involved information is integrated depends on the particular annotation: Some are variant-specific, some depend on the position or segment of the reference genome the variant is located at, while others are independent of genomic location and based only on the type of nucleotide exchange that occurs.

Inclusive and exclusive annotation management

Annotations can be managed inclusively or exclusively. Exclusive means that any information from an annotation may be considered only within the context of that annotation, in contrast to inclusive management that also considers what is known from other annotations. For example, any stop-gain mutation by itself would normally be considered highly pathogenic. However, if we put that stop-gain mutation in the context of the associated transcript, we may find that the new stop codon is only one amino acid prior to the previous stop codon, or that the involved transcript is rarely expressed at all. Similarly, we may observe a highly conserved sequence position in an intron of one gene and wonder about the conservation of that location (Fig. 18). If we find that the same genomic locus is also part of the exon of another gene and causes a synonymous effect there, that could explain the high conservation and lead to a more benign assessment of the variant.

Whether different annotations are managed inclusively or exclusively depends on the model. We have found that it makes sense to treat gene annotation, i.e. multiple genes annotated to one site, exclusively while all other annotations should be treated inclusively. In that case, it is important that only a single gene is annotated per variant during training so as not to prioritize one variant with multiple genes nearby over others with only one gene. Further, there has to be a heuristic on how to set precedence

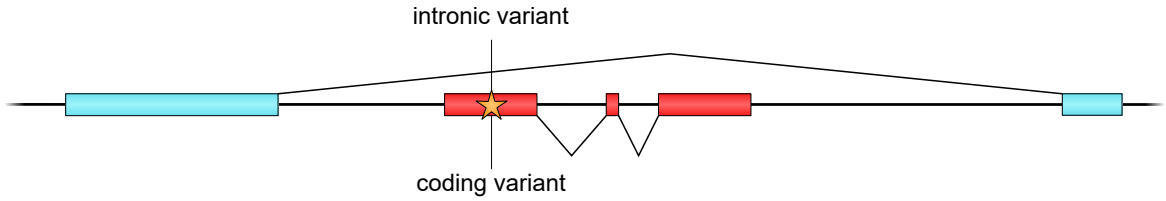

Figure 18: **Multiple consequences for one variant.** There are many cases of partial or complete overlap between different genes in the human genome. Variants in those regions may be associated with all genes, possibly with different consequences, as illustrated by the example of an intronic variant in the blue gene that is also a coding variant in the red gene. While changes in the DNA affect all genes, sequence constraints may originate from only one or multiple genes.

between exclusive predictions so that the synonymous variant of one gene is predicted preferentially over the non-coding element of another gene.

Precomputed and ad hoc data

Most variant annotations are derived from genome-wide data sources. Time-consuming processing steps, such as converting raw data to the correct format, are performed once for all possible variants and only parsed and possibly adjusted during annotation. The data is stored in some sort of relational database, so retrieving the annotation for a variant requires a single look-up in the database index. For some annotations, there exists no previously computed (precomputed) data. A good example of such annotations are those that depend on other machine learning models, e.g splicing prediction via deep neural networks. In that case, it may have been considered too time-consuming to generate the annotation once genome-wide. Even if it later proves necessary to do just that, as long as the annotation is explored only as a possible model feature, it can be integrated into the model as *ad hoc* annotation. *Ad hoc* annotations are generated only for the variants that are needed, which entails the limitation that the annotation process thus requires more time and resources. However, they also bring benefits, since dependencies like the gene annotation build can be adapted more easily than with precomputed annotations.

Annotating insertions and deletions

Assigning annotations for InDels is in many cases more complicated than for SNVs. Due to the many different possible InDels at each genomic position, precomputed annotations are often only available for SNVs. Annotation of InDels is further complicated by the fact that they are not restricted to a single nucleotide position in the genome. Where SNVs are annotated with the value at that position, insertions in principle do not affect any position in the existing sequence, while deletions are spread over multiple positions. As a compromise, we retrieve all position values for deletions and those before and after for insertions (Fig. 19). A selection such as minimum, maximum or average may then be applied to reduce all to a single value. Depending on the data type, the resulting value distribution may be significantly different from that of SNVs. Careful consideration must therefore be given to the extent to which InDel annotations and the resulting variant effect scores are comparable to those of SNVs.

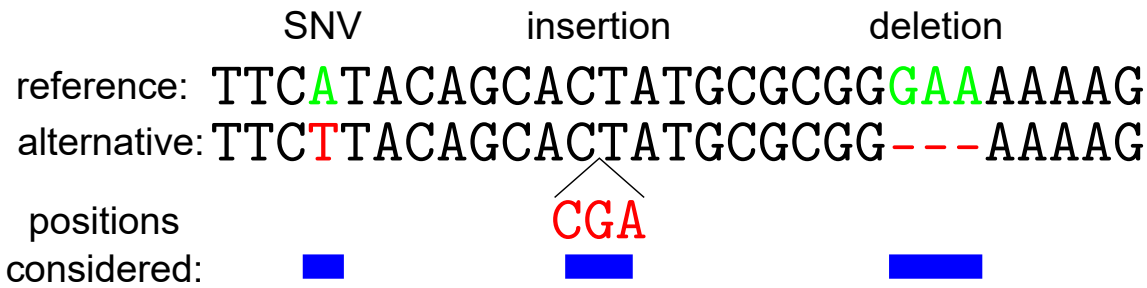

Figure 19: **Positions of annotation considered per variant type.** SNVs are simple to annotate, as only the value at the position of the variant is considered. For deletions, this is extended to all deleted nucleotides. Insertions are annotated from the positions immediately before and after the inserted sequence.

Gain and loss of function

Sequence changes can affect many different processes, like the binding of a transcription factor or the activity of a protein domain. For the final variant effect score, significance does not depend on the general direction of that effect, i.e. if there is a gain of something that did not exist before or the loss of something that did. However, depending on how they are generated, most annotations are limited to loss of function, as existing, observable processes or the observable state of the genome are described. A change in this *status quo* is interpreted in the sense that the annotated process, e.g. binding

of a protein or expression of a transcript, is not working anymore. In contrast, the reverse case, e.g. binding of a new protein or expression of a novel transcript, is much more difficult to predict. In cases where gaining of activity at a site is predicted, the uncertainty about the gained activity is often different from that of a predicted loss of activity. Even if the gain is the result of a variant, the further consequences of that gain are typically unknown. Any novel activity, when considered in terms of evolutionary fitness or pathogenicity, may have no, only negative, only positive, or a mix of different consequences. This distinguishes it from the existing activity, which is assumed to be predominantly positive because otherwise the respective mechanism would have been long been selected against.

We have relied on two different strategies for coping with different effect sizes in gain and loss predictions: Either positive and negative effects are mapped into separate features or limits are imposed so that any gain effects are masked. In this way, weights for the different directions of consequence prediction are offset to the machine learning model. Nevertheless, gain of function predictions remain a limiting factor for any variant effect score.

Reference genomes and gene annotations

Many genomic annotations are based on sequencing data that has been mapped to the reference genome. Similarly, transcriptomic data is matched to a set of transcript annotations from a gene annotation build. With progress in genome research, these references are continuously updated. One of the key ideas of open science is reproducibility. For annotations, it enables the reprocessing of existing raw data based on another reference, which may be used to train a classifier with the same parameters, but for example on another genome build. Unfortunately, the raw sequencing data is often not readily available, or reprocessing takes, at minimum, a lot of effort. In that case, annotations may be lifted between genome builds via a coordinate transfer. This liftover process is not perfect as new genome builds have historically changed more than just coordinates of variants, with the consequence that lifted annotations are missing in crucial and especially novel sequences of the genome. Nevertheless, lifted annotations are important for models that are compared across different reference sequences. The same applies to gene annotations, as the same build should be used for all annotations but is often not available, especially when annotations are precomputed.

Data precision

The level of detail that is available from a data source is often higher than needed for interpretation. Since even in the age of big data, storage space is always limited, the level of precision inflates the size of the annotation file set and files of annotated, scored variants, often without impact on the final model score. For example, if one very important annotation has a precision of 0.01 but is multiplied with a coefficient of 0.1, the uncertainty generated by that annotation is $0.001 = 10^{-3}$. If another annotation has a much higher level of precision, i.e. an annotation value of 1.2345 that is multiplied in a linear model with a coefficient of 0.03, the influence on the total score of the last value digit is $0.0005 \times 0.03 = 0.000015 = 1.5 \times 10^{-5}$. As the final score uncertainty is already limited by the precision of the first annotation, the increased precision of the second may be omitted. There is no evidence that reducing the precision of some annotations does improve model performance. However, for text-based files such as any tabix accessed format, a shorter value string leads to a reduction in size of the final data file. For example, we were able to reduce the size of the archive with all annotations from 194 GB in CADD GRCh38-v1.4 to 168 GB in GRCh38-v1.5 by reducing the precision of ChIP-seq annotations derived from ENCODE. We note that other data formats may reduce file sizes even more. For example, the genome-wide file containing CADD scores of all possible SNVs in the gzip-compressed TSV format is 80 GB, while the same data in the bigWig format is only 40 GB. Hence, reduction of precision is only a last resort but requires no further adjustments in processing.

4.3 Building the training matrix

Once all information about a variant is collected, that information has to be put into context before it can be evaluated by a machine learning model. The various data formats, strings, categories, and numbers need to be converted into a numeric vector. Some of the generated features are dependent on a single annotation while others are a combination of multiple sources. How features are prepared, transformed, combined, encoded, and imputed is explored via feature engineering. Which features are used in the model is then decided by feature selection.

Missing values, undefined or undetected?

There are many reasons why an annotation may not be available for one particular variant. Besides when they are not applicable, i.e. the affected amino acid of a non-coding SNV, this situation often occurs when that annotation is missing in some regions of the genome. Since the final feature vector of each variant must have the same dimensionality, the missing feature is then imputed with a replacement value.

How undefined values are selected has to be decided carefully. For example, a conservation score may be missing because no other species genome could be aligned to that genomic region. In that situation, it is unclear whether that genomic segment is of very low conservation and therefore not part of the other genomes, or whether that particular genomic region is not assembled in other genomes. If it is not possible to decide whether the annotation is undefined or undetected, multiple situations that have totally different reasons for missing an annotation may be considered jointly. These uncertainties can be magnified when multiple annotations depend on a mutual basis, i.e. different conservation scores on multiple sequence alignments.

The probably best solution to minimize the effect of missing annotations is to exclude some genomic regions where model scoring is unreliable, for example, alternative haplotypes, the mitochondrial genome, or chromosome Y. While it may be annoying for score users to be missing scores for some variants, score artifacts in these regions may lead to undesired results. For all other variants, it is possible to distinguish between variants that are missing a value and all other variants by introducing an additional feature: A binary indicator that states whether a particular annotation is defined for that variant or not.

Feature interactions

Feature interactions are features that depend on multiple different annotations which get combined in various ways. An example is the feature-cross where the value of a numeric annotation is multiplied with the one-hot-encoded vector of a categorical annotation. Whether feature interactions are relevant for a model depends on the applied algorithm. While tree-based learners, as well as deep learning architectures, will combine features in many different ways, linear models may benefit from carefully crafted feature combinations. Depending on the selection of the training set, feature interactions have to be considered carefully as they may identify variant categories in ways not obvious from the single annotations.

Feature correlation and degrees of freedom

Many features of similar definition are based on common sources. A good example are multiple sequence alignments that are used to generate different conservation scores, i.e. based on primate versus based on mammalian genomes. Naturally, these are significantly correlated. One reason is that a gene that has been evolutionary stable in the short term is likely to be stable in the long term. Further, the genomes found in the primate alignment may also be part of the mammalian alignment. Even if this were not the case, the evolutionary history of all species would be shared. In any case, the correlation makes it impossible to attribute the predicted score to a single annotation and may even lead to model instability when coefficients of different annotations compensate each other. One possible solution for annotation correlation is dimension reduction via methods such as clustering or PCA, although a resulting loss of information may limit the use of some annotations.

Degrees of freedom that are introduced with feature encoding have to be considered as well. A prime example for this are nucleotide exchanges: Ignoring the more complex nucleotide exchanges of InDels, there are several different encodings for a SNV. From four nucleotides, each possibly substituted for three other nucleotides, there are 12 nucleotide exchanges. The exchange A(denine) to G(uanine) on the + strand is the same as T(hymidine) to C(ytosine) on the - strand of the genome. If the strand is not considered separately, the 12 theoretical nucleotide exchanges, therefore, reduce to six. An even simpler reduction is the classification of mutations in transversions and transitions. Encoding both, the exact nucleotide exchange and whether a mutation is a transversion is adding an additional degree of freedom to the model. Amino acid scores can be similarly overdetermined, with substitution scores like Grantham (Grantham, 1974) or PAM (Dayhoff *et al.*, 1979) acting as complex encodings that provide a weighting of every amino acid substitution according to coefficients of each substitution. They may nevertheless be useful as they provide an initialization weight for each substitution and are minimizing regularization strength for rare events that would need large coefficients without them.

Storing the data as a sparse matrix

With hundreds of features and potentially millions of variants, a variant scoring matrix can be quite large. One of the main drivers of matrix size are one-hot-encoded annotations and feature-crosses that lead to many single variables per annotation. Due to the

encoding limiting the actual value to only one of these variables, most variables of the obtained matrix will be zero, meaning that the annotation is not of that value. The size of such a matrix may be reduced via compression. However, machine learning on compressed data is complicated, slow, and not available on most frameworks. Instead, the data may be converted to a sparse matrix format. Sparse matrix notations only store values different from zero, which significantly reduces the size of any matrix with many empty variables. The obtained files enable fast data loading and training.

4.4 Training the model

After annotation and feature design, the training data set is loaded and model training is started. Like any other machine learning problem, training a variant scoring model can be optimized in terms of algorithm and hyperparameter settings. During training, the convergence of the model is monitored in order to prevent overfitting. Especially relevant to variant scoring is how the training set influences validation and testing. The common definition of a fraction of the training set as hold-out for testing can lead to biases for certain genomic regions. For example, using the popular ClinVar data set for training will include many variants from the gene BRCA1 in the pathogenic class of any training set. That creates the possibility that any variant in that gene is classified as pathogenic. With training and testing variants sharing annotations, a good performance in that case may be due to being able to differentiate between important and less important genes rather than variants. Instead, gene and chromosome hold-outs use all variants from certain regions as benchmark sets, reducing the risk that observed good results are due to overlap in training and validation.

Any variant data set that is not the entire genome has limits and biases. Besides generalization issues, i.e. amino acid effect predictor being used for splice site effects, the main challenge in this regard are observation biases: Presently known pathogenic variants are usually known to be pathogenic because the involved genes have been investigated thoroughly. This investigation may have happened because of other variants in the same gene, due to an associated disease, homologous genes in other species, or because of technical limitations like gene length. In order to show that the model is not only learning these biases but able to extrapolate to other data, requires validation with independent test data sets. Many of these alternative data set may be classified by another metric, i.e. some functional assay that is only associated with pathogenicity. In that case, relative enrichment is used to show that a model is better than random (or

another model) at predicting that metric.

4.5 Delivering a variant score

Once a variant scoring model is generated, it is time to make it public. Besides the academic publication, this means enabling other people to score variants as a service. While the target audience of a variant score at this stage are other academic researchers, we should neither expect the technical expertise of a computational biologist nor the effort of reading long and complicated documentation to use the service. This includes the necessary interaction to obtain the score value for a variant, as well as providing context so that the score it is not just a number in a given interval and can be interpreted in human categories such as pathogenic or benign.

The arguably most practical service is a public web server, where a variant file in a common data format such as VCF can be uploaded. The web server then performs its prediction on the parsed variants and returns the score as either a dynamic web page or a file download. We expect that the entire process is fast and reliable. Speedup may be achieved via prescoring of variant sets that are found repeatedly, i.e. genome-wide scoring of SNVs. In any case, it is important that the score of each variant remains reproducible at all times, and is independent of the uploaded file or server status.

In addition to the web server, we ideally want to provide the source code that produces the variant score. The most compelling argument to make the source available is that it enables further research in the area, as other scientists can modify and extend the model. Beyond that, it makes the entire process transparent and removes the pressure to rely on the availability, stability, and constraints of the web server.

Genome-wide score calculation

Instead of annotating and scoring genomic variants only when requested, it is possible to prescore variants: Generating all scores at once for a large set of variants and then retrieving only those from the stored file that are requested. Prescoring has three key advantages: First, every variant is scored only once. Second, summarizing all variant annotation requests in a single access means that the annotation process can be run continuously. Then, annotation requests will not result in random file accesses but a line-by-line parsing of all data in the annotation files. Third, any provided service becomes much faster as each score request is reduced to a single random file access on the prescored file.

Genome-wide scoring can be routinely applied to predict all SNVs in a genome, with three possible SNVs for any nucleotide in the genome sequence. While theoretically possible for deletions, where for each nucleotide in the genome there is one deletion of each length, there is an almost unlimited number of insertions as the inserted sequences may be any possible combination of the four nucleotides. As such, prescoring of InDels is better limited to variants that are relatively likely to be found, for example, due to high allele frequency or because observed in popular genes or data sets.

Phred-scores

Scoring SNVs genome-wide provides a good overview of possible score values. The entire distribution can be analyzed statistically and enables ranking of every single variant in comparison to all SNVs. Since the rank of each variant is a large number that can be hard to grasp, the rank is modified via a comparative descriptor like an empirical p-value or log transformation of the relative position. One popular log transformation is Phred-scaling (Ewing & Green, 1998). From the relative rank of each variant $r = rank(\text{RawScore})$, a Phred-score is calculated via:

$$\text{Phred-score} = -10 \log_{10}(\frac{r}{\text{total number of SNVs}})$$

Any relative rank-based evaluation is dependent on the number of instances ranked.

Depending on whether more or fewer segments of the genome are included in the genome-wide calculation, i.e. from alternative haplotypes or previously uncharacterized regions like centromeres, the relative changes in rank lead to a shift in Phred-score. Hence, the number and score distribution of all SNVs is determined once and used to generate a conversion table. The finalized table can be quickly applied to all genomic variants including InDels to associate raw model and Phred-scores.

The interpretation of a Phred-score is subjective and it is impossible to define the one threshold above which variants are generally considered as pathogenic. Since different probabilities in training and test set lead to a relative label shift and considering the different stages of variant assessment, the selection of a threshold is influenced by the number and type of variants (Guo *et al.*, 2017; Alexandari *et al.*, 2020). For CADD, users have been using thresholds like Phred-score > 10, 15 or 20 (Itan *et al.*, 2016), although all of these correspond to arbitrary values and have most likely been chosen for the beauty of their numeric values.

Accessibility

The final variant score data has to be accessible to many different people. Unfortunately, there are many different data formats in bioinformatics that, though mostly convertible, generate adoption obstacles. For the single application, this can only be minimized by implementing file standards and linking format converters.

Aside from the data format, accessibility raises the question of how data is obtained by a user. For computational biologists, using a command-line interface is very common, and they may prefer a local database or an API from which variants are retrieved. In contrast, someone without a computational background may prefer a web service where a file with variants is uploaded and the result is downloaded. As speed is an important aspect of the retrieval process, prescoring of frequently requested variants may reduce the average processing time and enables new applications like browser tracks for the UCSC genome browser (Kent *et al.*, 2002). It should be noted though, that prescoring can also misguide users. We have found that available resources of prescored variants, such as the popular variant score library dbNSFP (Liu *et al.*, 2016), can lead to the conclusion that some variants cannot be scored with a variant score, simply because it is not included with the other prescored variants. While such confusion is probably unavoidable, these shortcomings should be publicly documented in any prescored resource.

Data protection

Finally, genomic data is highly informative of the genomes it is derived from and accordingly highly protected in various jurisdictions around the world. To retain full control of the involved personal data, any user of the variant score should prefer offline scoring of all variants on a computer system that is under their control. However, this is not always possible as the required resources to run variant scoring can be enormous and are, even if not making it impossible, slowing down the application for the user. To protect personal data, the provided web services should store only strictly necessary data and not share it with third parties. Transparency about the collected data increases trust and minimizes confusion regarding whether the user is legally allowed to use a service or not.

5 Implementing a new variant scoring workflow

The model training and variant scoring workflow that was developed with a focus on the variant classifier CADD, is mainly implemented in four Snakemake pipelines (Fig. 20). All pipelines automatically manage dependencies via conda and are adaptable to novel annotations, training, and validation sets via configuration files. Annotations and features are implemented as dedicated object hierarchies that manage the parsing of common data types and formats while enabling the custom integration of many kinds of variant information. While the entire workflow is adapted to the CADD project, e.g. large size and particularities of the training data set, it is able to train any supervised machine learning based variant classification independent of training set, annotations, model algorithm, or application scenario.

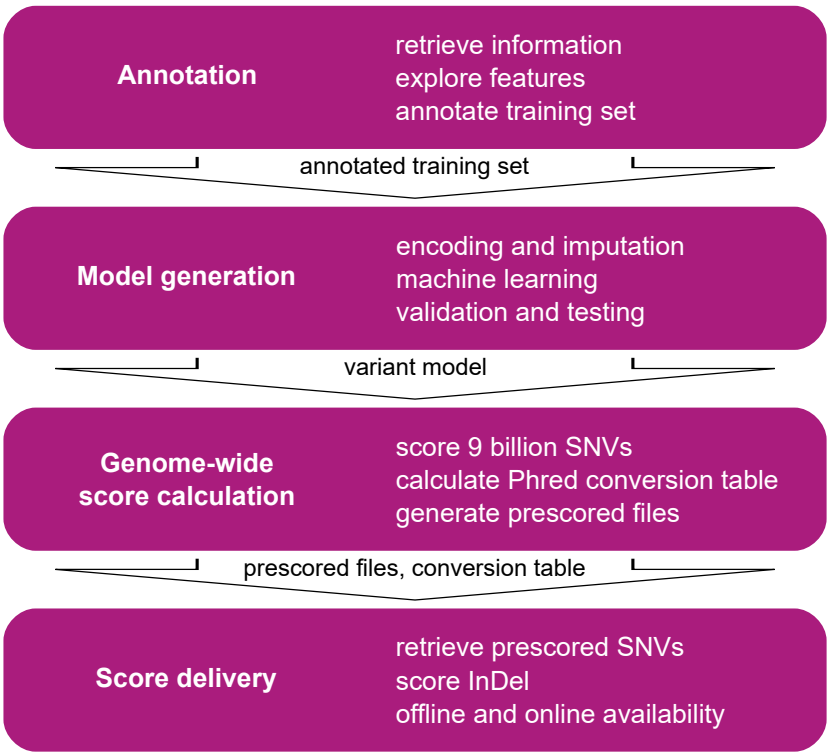

Figure 20: **Variant scoring pipelines.** In total four pipelines execute all necessary steps from the initial variant set to the scored variant. The pipelines are executed one after the other.

The first two pipelines implement the annotation of the training data set and model training. These two steps are kept separately as the former is a relatively slow process that is ideally executed once to annotate the entire training set with all available

annotations. It is supplemented by an AddAnnotation pipeline to later add further annotations to the large files of annotated variants. Model training follows as a faster and more flexible process that enables rapid and parallel prototyping of feature combinations and preprocessing, hyperparameter optimization, and model benchmarking with different testing and validation data sets. The third pipeline is scoring all SNVs genome-wide and generates the Phred-score conversion table. Finally, the fourth pipeline is able to score variants from provided VCF files. It is used as an online service, to score variants on the CADD web server, and as an offline service, that enables users to score variants on their own system.

5.1 Training set annotation

The pipeline to annotate the training set (Fig. 21) is set up to process millions of variants, including SNVs and InDels, that are spread through the entire human genome. As variants are considered independent of each other, data parallelization is achieved by a simple split of the input into sets of fixed size. Variant annotation itself is split into three steps: annotation with Ensembl VEP, *ad hoc* annotation with other VCF-annotation scripts, and the final summary of all information into a tab-separated-value (TSV) text file that includes further annotations from external data files. The entire process is managed via a general configuration file that specifies the training set, VEP parameter settings, and extensions, the *ad hoc* scripts that are annotating based on VCF files, the split size used for parallelization, and a second configuration file, AnnotationConfig, that specifies all annotation columns in the output.

Most variant annotations that depend on the gene annotation are retrieved from Ensembl via the Variant Effect Predictor (VEP). These annotations include gene and transcript identifiers which are further analyzed to predict the position and consequence of a particular variant in a gene, e.g. a missense variant causing an exchange of the amino acid leucine to arginine. For variants with multiple overlapping or nearby genes, VEP annotates one possible consequence per variant per gene. VEP also provides gene annotation build based annotations like PolyPhen-2 and SIFT. In the pipeline, genome build, Ensembl database version, as well as VEP extensions, are managed in the configuration file. VEP is run in as an offline script installed from bioconda (Grüning *et al.*, 2018) using VCF output.

The VEP annotated VCF file can be extended in the second annotation step by further VCF-annotating scripts. Separating this process from the other annotation steps

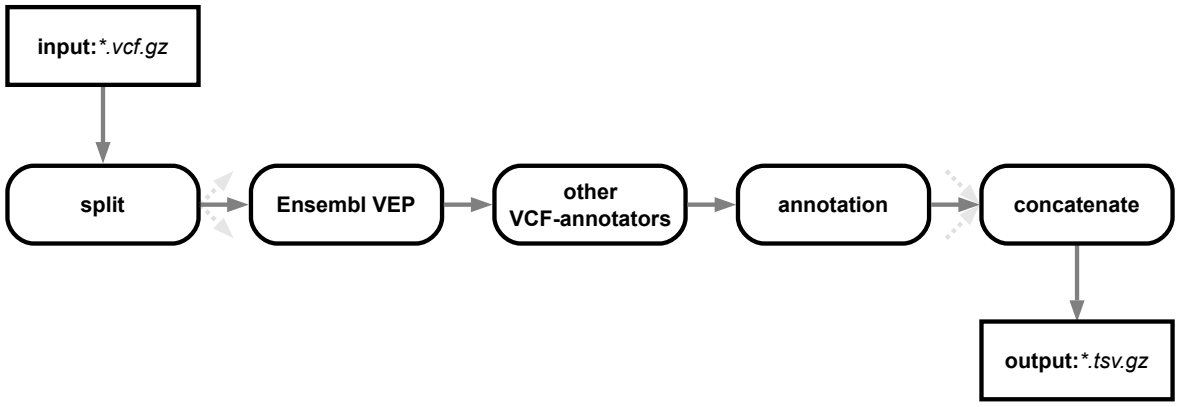

Figure 21: **Rule graph of annotation pipeline.** VCF files are annotated in three steps: Ensembl VEP, other VCF-annotation scripts and the final annotation that converts the variants from VCF to a column-wise tab-separated format. Data parallelization divides the main processing steps through a split of the input file. The results are concatenated after all annotation steps have been executed.

has the advantage that it can be run in a more exploratory setting, using *ad hoc* annotation generation that depends on custom libraries that may differ from the rest of the pipeline. As such, the step was established to run deep neural networks like MMSplice and SpliceAI to annotate variants with splicing predictions. In general, we have found it to be advantageous to calculate any annotation genome-wide once so that each position or variant has to be covered only a single time. However, genome-wide score calculation is computationally expensive, especially so for complex deep neural networks. The VCF annotation step serves as an intermediate step for *ad hoc* annotations, as the script is only used to annotate the training set and explore potential features. If the annotation is later integrated into the model, the annotation is calculated genome-wide and moved to the third annotation step.

The third step uses the annotated VCF file as input and converts it into a gzip-compressed tab-separated text file. Each column in the file is one annotation field while each row corresponds to a variant. This file format has the advantage that it is human-readable as well as simple to parse and extend. Which annotation fields are included is controlled by a configuration file, named AnnotationConfig. AnnotationConfig is a text file in python configParser format that contains two sections: The first section defines the global path to a root directory from where all annotation data files are localized and specifies the FASTA (Lipman & Pearson, 1985) genome reference file. The second section

lists all the annotations that are to be included, specifying the respective annotation data file for those annotations absent from the VCF.

The annotations themselves are defined as classes in the python library Annotation.py (Sc. 2) that is part of the variant scoring workflow. It specifies for each annotation class whether the annotation depends on a data file and how that file is accessed, whether the annotation depends on the variant consequence and how many annotation columns are provided by the annotation. To keep this source code structured and maintainable, the different annotation classes inherit common functions like file access and InDel scoring from meta-classes. Most data accesses are processed with SAMtools and tabix. This has the advantage that compressed data files can be reused without duplication in different configurations while the annotation process benefits from the fast random file access enabled by the tabix index. The library is designed as a list of all annotations so that further annotations can be added.

Sample code 2: Definition of annotations in Annotation.py (simplified)

```
class Annotation(object):
    consequence = False
    mandatory = False

class FeatureAnnotation(Annotation):
    def process(self, res):
        self._retrieve(res)
        return self._get_score(res)

class TabixAnnotation(FeatureAnnotation):
    multirange = False
    rangescore = False
    zerobased = False

    def load(self, args):
        if self.path != '' and (not os.path.exists(self.path)):
            sys.stderr.write("%s:␣Require␣valid␣path" % self.name)
        elif os.path.exists(self.path):
            self.tabix = pysam.TabixFile(self.path,'r'), self.name
            self.continuous = args.continuous

    def _retrieve(self, res):
        self.tabix = get_range_from_tabix(self.tabix,
                                          res['Chrom'],
                                          res['Start'],
                                          res['End'],
                                          rangescore=self.rangescore,
                                          continuous=self.continuous,
                                          multirange=self.multirange,
                                          zerobased=self.zerobased)
        self.score = self.tabix[6]
```

```
class ScoreHighest():
    def _get_score(self, res):
        if len(self.score) == 1: res[self.name] = self.score[0][-1]
        elif len(self.score) > 1:
            helper = map(lambda x:(float(x[-1]),x[-1]),self.score)
            helper.sort()
            res[self.name] = helper[-1][-1]
        return res

class PolyPhen(Annotation):
    name = 'PolyPhen'
    features = ['PolyPhenCat', 'PolyPhenVal']
    consequence = True

    def process(self, res):
        if res['PolyPhen'] != '':
            cat, val = res['PolyPhen'].split('(')
            res['PolyPhenVal'] = val[:-1] # remove closing ')'
            res['PolyPhenCat'] = cat
        return res

class PriPhCons(TabixAnnotation, ScoreHighest):
    name = 'priPhCons'
    path = '/phastCons/primates_nohuman.tsv.gz'
```

The annotation script parses the AnnotationConfig, initiates all annotations as Annotation objects, and checks the availability of the data sources. Variants from the second step are read iteratively as all annotations are processed into the row of annotation fields. For variants where more than one possible consequence is found by VEP, separate variant annotation rows are created. Whether all exclusive annotations per variant are kept in the final output, or only one consequence is retained, is specified in the overall configuration file. The annotated files are finally concatenated and saved for model training.

5.2 Model training

The model training pipeline takes the annotated training data set, selects features from those annotations, and trains one or more machine learning classifiers. The trained variant scoring models are then benchmarked on several independent test sets. All necessary settings are configured via a configuration file. Though the pipeline is adapted to training models with scikit-learn, the common structure makes it possible to train models with other machine learning frameworks like GraphLab (Low *et al.*, 2014), TensorFlow (Abadi *et al.*, 2015) or Keras (Chollet *et al.*, 2015) with few changes. The pipeline consists of three structural parts with several rules each: (1) preparation of the training set including encoding, imputation, and conversion into a sparse matrix, (2) model training and (3) benchmarking that annotates and scores validation and testing data set. As described before for the annotation pipeline, data parallelization is implemented for involved rules (Fig. 22) including training set encoding and test data set annotation by a split of the variant set. Similarly, multiple machine learning models may be trained at the same time. Before model training, the training matrix is stored in sparse matrix notation. This enables very large training matrices with several hundred features and millions of instances, that would require multiple hundreds of GB if stored in a dense matrix.

The pipeline rules labeled as encoding cover more translating annotations into numeric features. The EncodingConfig file lists feature tracks that are to be included in the final model. Defined are feature tracks in the source code as a large python dictionary in the file tracks.py (Sc. 3). The EncodingConfig selects feature tracks from that file, enabling the reuse of features. Each feature track is a configuration item that defines how the variant annotation stored in a character string is converted to a numeric format for the machine learning algorithm. It specifies string to number conversions, one-hot-encoding of categorical variables, defines the imputation value if the annotation is undefined, and enables simple processing like transformations or deriving a feature from multiple annotations. Additional functions can specify changes to an annotation when a variant is inverted by replacing reference and alternative sequence, a feature that is important for training the variant classifier CADD. Finally, encoding adds class labels to the encoded training vector to enable supervised machine learning and may select a random hold-out to track hold-out validation accuracy during model training.

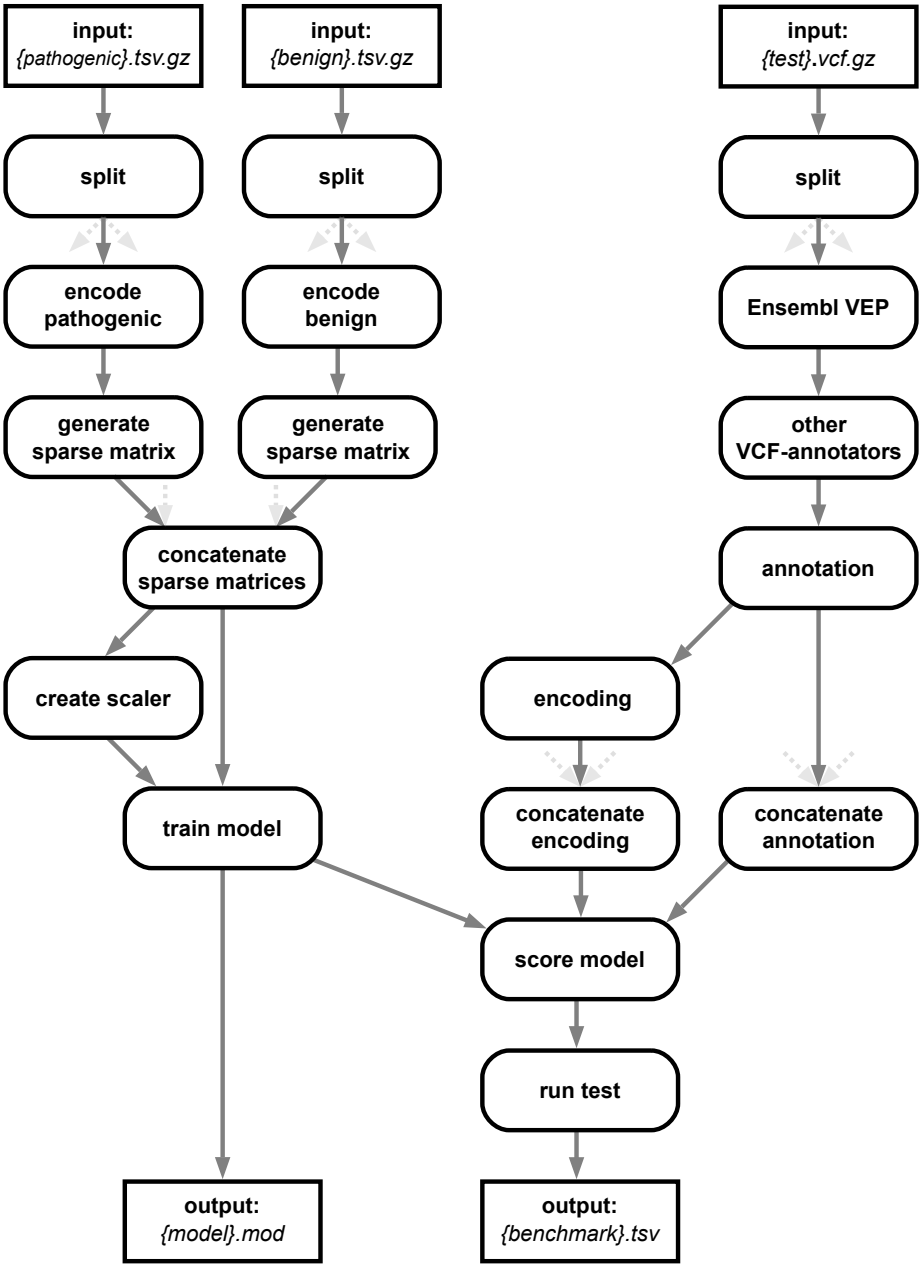

Figure 22: **Rule graph of training and benchmarking pipeline.** The annotated training data files are encoded in a single large sparse matrix (left). Test data is annotated and encoded like the training data but not converted into a sparse format (right). Depending on the type of benchmark and specified by a configuration file, multiple scored files (AUC benchmarks) or a metric given from the input file (correlation and Kruskal-Wallis benchmarks) are used for the test.

Sample code 3: Definition of feature tracks in tracks.py (adapted)

```
trackData = {
'ref': {
  'description': 'Reference␣allele',
  'type': list,
  'categories': ['A','C','G','T','N'],
  'dependencies': ['type', 'ref', 'alt'],
  'derive': lambda x: 'N' if x['type'] != 'SNV' else x['ref'],
  'na_value': 'N',
  'hcdiff_derive': lambda x: 'N' if x['type'] != 'SNV' else x['alt']
},
'length': {
  'description': 'Number␣of␣inserted/deleted␣bases',
  'type': int,
  'transformation': lambda x: min(x, 49)
},
'priphcons': {
  'description': 'Primate␣PhastCons␣conservation␣score',
  'type': float,
  'na_value': 0.115,
},
...
}
```

The encoded data is output in a comma-separated-value (CSV) file, that is subsequently converted into a sparse matrix. For sparse encoding, the pipeline uses the SciPy (`https://scipy.org`) implementation of compressed sparse row representation (CSR, Fig. 23). The CSR format enables fast concatenation of rows (vstack), that is used to merge sparse variant matrices, that where generated in parallel, into one large training matrix.

$$\begin{pmatrix} 0 & 5 & 7 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 8 & 6 & 0 & 6 \\ 3 & 0 & 1 & 0 & 2 & 0 & 0 \\ 0 & 0 & 4 & 0 & 5 & 0 & 0 \end{pmatrix} = \begin{array}{l} \texttt{data\_vector = [5,7,2,1,8,6,6,3,1,2,4,5]} \\ \texttt{row\_index = [0,4,7,10]} \\ \texttt{column\_index = [1,2,4,5,3,4,6,0,2,4,2,4]} \end{array}$$

Figure 23: **CSR matrix notation.** Compared to the dense notation (left), the compressed sparse row format (CSR, right) stores the data in three vectors: data vector, row index and column index. The data vector stores all non-zero values from the matrix as 32-bit floating point numbers. The row index stores for each instance (row of the matrix) the position of values that are non-zero in the column index and data vector. The column index stores the column numbers for all values in the data vector.

Model training starts with scaling of the feature matrix. By default, the pipeline uses the StandardScaler from scikit-learn that transforms each feature to the same variance but does not shift the mean to zero to retain a sparse data matrix. Model hyperparameters are specified via a separate ModelConfig, that enables exploring different machine learning algorithms and hyperparameter optimization. The trained model object is stored in conjunction with the scaler in a joblib file (`https://joblib.readthedocs.io`).

To evaluate different model settings for validation and testing purposes, the pipeline is able to benchmark the trained models on selected test cases. Besides accuracy on the hold-out that is logged during model training, these test cases are defined in the TestConfig file. Supported are correlation, area under the curve (AUC), and Kruskal-Wallis tests. Correlation is used to compare prediction scores from one set against another metric, AUC to benchmark the ability of a model to distinguish between two variant sets, and similarly, the Kruskal-Wallis test to evaluate the ability to distinguish an arbitrary number of categories. The obtain the benchmark scores, variants are annotated and encoded via the same steps as the training data set. Variants are then scored with the trained models. Depending on the defined test metric, the scores are finally compared against another set or values previously defined as a separate column in the input file.

While the annotation of a large training set uses hundreds of compute nodes for hours, the model training pipeline can be run with fewer computational resources. Based on the different configuration files, i.e. EncodingConfig, ModelConfig, and TestConfig, and an already annotated training data set, it enables rapid exploration of different feature sets and model hyperparameters. The final model may be chosen from the benchmark results and is used in the further steps for variant scoring.

5.3 Overview of performance benchmarks

In most machine learning settings, it is a central objective to predict instances in a hold-out of the training set accurately. CADD is an exception to this rule. The training set of human-derived versus simulated variants serves only as a surrogate for the real objective of the classifier: to detect pathogenic variants. While pathogenic variants are expected to be depleted in the set of human-derived variants, there are many other data sets that, while overall not as large and extensive as the CADD training set, are better suited for model benchmarking.

In an ongoing progress over multiple years, more than 15 different benchmarks have been adapted in the development of different CADD versions. All adapted benchmarks

fall into three main categories: (I) two-class comparisons measuring Area Under the Curve (AUC) between class predictions, (II) correlations of the model score with an assay defined metric, and (III) multiclass comparison based on non-parametric statistical tests.

Correlation with allele frequencies

The allele frequencies of variants observed in population studies are used as a surrogate for pathogenicity. Similar to the CADD training set of human-derived variants, frequent variants are less likely to be pathogenic due to purifying selection. Variant allele frequencies are obtained from the 1000 Genomes project (The 1000 Genomes Project Consortium, 2015). Separate for SNVs and InDels, the allele frequencies of 100,000 randomly selected variants are correlated with model scores, optimizing for high correlations. Correlations on these benchmarks are generally very low (Spearman correlations between 0.03 and 0.05). This benchmark is especially valuable for InDels as there are few large benchmark data set that do not entirely consist of SNV.

ClinVar benchmarks

The largest currently publicly available curated database of human genomic variants is ClinVar (Landrum *et al.*, 2018). In ClinVar, variants are assigned to one of five clinical significance categories as defined by the American College of Medical Genetics and Genomics and the Association for Molecular Pathology (Richards *et al.*, 2015). The four significance categories are "pathogenic", "likely-pathogenic", "likely-benign" and "benign", with a fifth category "uncertain significance" when none of the other categories can be assigned. Assignments of variants are contributed from genetics laboratories worldwide. It should be noted that ClinVar assignment can be wrong and findings from different studies may have contradicting results. Some variants in the database are therefore assigned to multiple classes. A review status has been established for all ClinVar variants to address uncertainty in the classification but is not used here.

Variants that are classified in ClinVar categories are used for two-class benchmarks. The most obvious of multiple possible combinations is a comparison of the most and least severe categories, i.e. pathogenic versus benign. An alternative benchmark compares the categories likely-pathogenic and likely-benign. Limited to SNVs and short InDels, each of these categories contains multiple thousand variants. All include diverse sets of variants from coding and non-coding regions, that can be further split into missense, nonsense,

splicing, and intronic subsets that serve as valuable benchmarks for the respective variant types. auROC is the preferred metric in all ClinVar benchmarks as the variant numbers in positive and negative sets are not expected to represent the true fractions of pathogenic and neutral variants.

It is not entirely clear how well variants from the ClinVar categories benign and likely-benign represent neutral variants. Besides the consideration that the absence of evidence for pathogenicity is no evidence for absence of the latter (Richards *et al.*, 2015), all variants in ClinVar have been under selection bias. While an entire human genome contains millions of variants, some reason has driven a human analysis to investigate exactly those variants to a point that justified submission to ClinVar. Only about 10% of all human genes have been thoroughly researched (Edwards *et al.*, 2011) with the studied genes not being representative for all genes (Stoeger *et al.*, 2018). An example of a strongly investigated gene is the tumor suppressor gene BRCA1, which, as of ClinVar release from April 4, 2017, is annotated to 799 of 33,957 pathogenic and 230 of 10,500 benign variants in the database.

A substitute for the neutral set in the benchmark are variants that are found frequently in a population. As stated previously, frequent variants are thought to be under purifying selection and accordingly be mostly neutral. However, while pathogenic ClinVar variants are overwhelmingly found in coding regions, frequent variants are distributed throughout the entire human genome. Results from exome sequencing (Ng *et al.*, 2009) may be used to regionally limit population variants, with the benefit that more data is available today from exome than whole-genome sequencing. In this study, all variants with an allele frequency greater than 5% but lower than 50% from the ExAC (Exome Aggregation Consortium *et al.*, 2016) project are used as the neutral set.

Each of the three comparisons described, (A) ClinVar pathogenic versus ClinVar benign, (B) ClinVar pathogenic versus frequent variants from ExAC, and (C) ClinVar likely-pathogenic versus ClinVar likely-benign has certain disadvantages. The neutral set from B is certainly the least biased and distributed over the entire exome. However, the auROC metric of various predictors on B is in the order of magnitude of 0.95, meaning that both sets can be separated almost perfectly. While this is great from an overall perspective, it means that differences between multiple models are very small and depend on the classification of a small fraction of the entire data set. In comparison, as of CADD v1.6, no model has reached an auROC better than 0.8 on benchmark A. Benchmark C depends on how much doubt is involved in the classification of the

variants in the likely categories. Generally, the involved data set has been used less to evaluate other variant effect scores that may lead to circularity between model testing and validation, with the limitation that this is not true for variants that have been re-assigned from the more distinct categories. Due to the high number of variants in both categories, it is nevertheless a valuable benchmark to assess genome-wide pathogenicity prediction.

Association with measured variant effects

An ideal benchmark would consist of an assay that analyzes all genomic variants without bias in a single biological readout and compares the assay results to the predicted scores. Unfortunately, the total size of the genome is so gigantic that it is unfeasible to study all possible variants with the currently available technology. The selection of regions that are supposedly "interesting" creates biases for which pathogenic or neutral variants may be found. Variant effect prediction scores that are solely assessed on the chosen, well-characterized variants may fall into the same bias and may therefore never find currently unknown effects. Multiplexed assays of variant effect (MAVE) are designed to partially solve this bias by selecting a single genomic segment and analyzing all possible variants in those segments, associating them with a single phenotypic variable (Starita *et al.*, 2017). MAVEs have been established for coding and non-coding regions. The analysis of protein-coding regions with deep mutational scanning (DMS) substitutes each amino acid in a segment with every other possible amino acid and measures the expressed proteins' properties like stability, enzymatic activity, or ligand binding. Each DMS assay is specific to first a single protein and second one biophysical assay that measures a protein property, which means that variant effects are only comparable between variants evaluated in the same protein and assay. To access the prediction of missense variants, DMS data was collected for the human protein-coding genes BRCA1, DLG4, YAP1 from Gray *et al.* (2018) and TPMT from Matreyek *et al.* (2018) and is correlated to the scores of the developed models. DMS data sets are specific to a single amino acid exchange. Due to the degeneracy of the genetic code (64 nucleotide triplets translate into 20 amino acids), in many cases, multiple nucleotide variants can be assigned to a single DMS variant. Here, only DMS variants that can be explained by a SNV are included, with one SNV selected randomly if multiple are able to achieve the amino acid exchange.

Similar to DMS, although totally different in processing, are Massively Parallel Reporter Assays (MPRA), a type of MAVE for non-coding variants. In a MPRA, a non-

coding element, like an enhancer or promoter sequence, is cloned in a minimal gene construct that transcribes into a short reporter mRNA. The idea behind this sequence construct is that the non-coding element influences the transcription of the gene-based on its sequence, as the expression of the latter is measured via RNA-Seq. Using an error-prone PCR that generates random nucleotide exchanges, all possible SNV of the sequence of the non-coding elements are generated and tested in parallel with the assay. The effect of each SNV is then measured as the difference in mRNA expression between alternative and reference sequence. For the development of CADD models, MPRA of enhancers from ALDOB, ECR11, HBB, LDLR, SORT1, and TERT (Patwardhan *et al.*, 2009, 2012; Kircher *et al.*, 2019) are evaluated by correlating the expression changes to the model scores.

A benchmark data set that covers RNA splicing effects is MFASS (Cheung *et al.*, 2019). The MFASS data is based on a MAVE that tested almost 30.000 SNV in more than 2.000 human exons for splicing changes. Based on the observed change in splicing levels, the data set is split into two groups: splice-disrupted variants (sdv) and not splice-disrupted variants (no-sdv). Classifiers are compared in this benchmark on how well they differentiate between the two groups. Due to the strong imbalance in the data set (1.050 sdv to 26.593 no-sdv) being a result of the assay, not the chosen variant set, the benchmark is measured via the precision-recall curve (PRC).

Two smaller and more specialized benchmarks consist of variants in the genes Tumor Protein P53 (TP53) and human beta globin (HBB). TP53 is a tumor suppressor gene that is mutated very often in cancer patients. The protein that is translated from TP53 (p53) plays an important role in managing cell death, proliferation, and survival, assigning TP53 mutations a key role in enabling the extensive growth of tumor cells (Kruiswijk *et al.*, 2015). While pathogenic variants are depleted in standing variation, function-altering variants are expected to be found more often than neutral variants in cancer patients. A data set providing variant counts of missense variants in TP53 has been published by (Bouaoun *et al.*, 2016). The counts are correlated with CADD model scores. The HBB benchmark set consists of variants that are associated with three disease severity levels of thalassemia and hemoglobinopathies (Giardine *et al.*, 2007). The model predictions from the three severity levels are evaluated via a Kruskal-Wallis one-way analysis of variance test.

There are many more potential benchmark data sets. For example, a growing number of DMS and MPRA data sets is continuously becoming available, like those collected in

MaveDB (Esposito *et al.*, 2019). While these certainly present a good source for extended model testing, they remain to be integrated into the testing system once better features become available for functional variant interpretation. Many other data sets like protein databases are under various forms of selection bias or, like HGMD, require a license agreement to be used for analyses. Nevertheless, the number of available sets, as well as the number of variants in existing sets, is continuously growing and will further improve both training and benchmarking of variant prediction scores.

Most of the benchmarks listed here have been used for the validation of CADD models. This includes the assessment of the different features used in the models, as well as model selection. Based on machine learning best practices to prevent overfitting, a few benchmarks were excluded from the validation stage and were only run once a new model version had been selected. These "hold-out" data sets also serve as an estimator of how well the model would perform on other, unseen data. For genome-wide assays, this is achieved via a per chromosome split into validation and test set. For example, the comparison of pathogenic ClinVar variants and frequent variants from gnomAD is split into variants on unevenly numbered chromosomes in the validation set and evenly numbered chromosomes plus the gonosomes in the test set.

5.4 Model selection

CADD models are optimized and ultimately selected based on performance in the previously described benchmarks. In total, 13 benchmarks are used to evaluate models in a grid search of regression penalty and training iteration parameters while another two benchmarks are hold-outs for the final model test. Due to the number of benchmarks that are implemented, usually no single model is the best in all of those. Instead, a good model is supposed to work well on most of them. The different benchmark results can be summarized in one overall score. Metrics and ranges of the individual results are normalized per benchmark by subtracting the mean of all models and dividing by the standard deviation. The normalized benchmark results are then weighted by how important each benchmark is considered to be and aggregated into a single score by multiplying each benchmark by its weight and calculating the sum over all benchmarks. These benchmark results per model are not absolute but relative to all other models in the comparison. Each score value is impacted by the model set explored, as for example a very bad performing model can considerably affect the mean and standard deviation of a benchmark. That means that smaller differences between all other models are weighted

to a lesser degree in the final summary. An alternative would be to develop a fixed score range for each benchmark based on performance. In conclusion, a summary benchmark score can serve as a guide in model selection and has been used in the evaluation and selection of CADD model parameters. Nevertheless, it is important to keep in mind that the selection of which benchmarks are used, as well as which model parameters are analyzed, influences which model scores best in each setting.

A good model is not selected only based on performance. Another model selection criterion is consistency between versions. While CADD scores from multiple versions are not supposed to be used in a single analysis, it is important that predictions for a single variant or element do not fluctuate largely between releases. Besides the confusion, this may lead users to involuntarily do multiple testing by probing multiple versions in hope of finding some evidence for a hypothesis. Hence, annotations are only removed based on evidence of overfitting or added based on improved prediction, and not because some benchmark returns a slightly better result. For the same reason, annotations are closely matched between genome builds (Chapter 6.4) and model parameters are adjusted only after some consideration.

5.5 Pipelines for genome-wide scoring and final application

Due to the large number of variants, the genome-wide annotation process is different from the previously described individual variant annotation. Instead of performing random file accesses with tabix to retrieve annotations for each variant, annotation data files are streamed consecutively. While the latter is not practical when subsequent variants are far away, in genome-wide scoring subsequent SNVs are only one base pair apart. Here, it turned out useful to compute all annotations from the second annotation step (VCF annotating script) before the genome-wide annotation. Aside from these optimizations, the pipeline for genome-wide annotation is identical to the training set annotation.

In order to generate a conversion table that links the raw model scores to Phred-scores, SNVs are encoded and scored with the model. The scores of all SNVs are concatenated and sorted from most to least pathogenic. A conversion table is generated from the Phred-transformation of ranks retrieved from the sorted file. Finally, the Phred-scores of prescored variants are obtained with a single look-up in that conversion table.

The fourth pipeline in the workflow is the scoring pipeline. The scoring pipeline is used to evaluate any genomic SNV, InDel, or multi-nucleotide substitution. Based on a provided VCF file, it generates the associated model scores and returns them with

or without annotations in a gzip-compressed TSV file. Variant scoring is following the same steps as described for benchmarking. In addition, a validation step at the beginning checks the format of each variant (e.g. minimal InDel description) and removes variants whose reference allele does not match the reference genome. To shorten the process, the pipeline can use prescored files, such as generated by genome-wide scoring of SNVs. All variants that are found in a prescored file are retrieved from there and only the remaining ones are annotated and scored.

5.6 Distributing CADD scores

CADD scores are primarily made available via the website `https://cadd.gs.washington.edu`. The data itself is hosted on a web server in the Department of Genome Sciences at the University of Washington, Seattle. With CADD release v1.4, a relaunch of the website moved all website traffic to `https` and considerably extended the services provided. As was possible prior to the release, the user is able to score variants from VCF files containing SNVs and short InDels online via the web server. A mirror of the web server is available at `https://cadd.bihealth.org`.

Alternatively, it is possible to download prescored variant sets for offline use, including the genome-wide scores of approximately nine billion potential SNVs. It was found that many users are interested only in a small number of candidate variants. For those, the score lookup process was simplified and accelerated through a new interface that provides scores and annotations for a single SNV, a genomic coordinate, or ranges thereof. Internally, this is processed via a tabix file access to the prescored genome-wide SNV files. The score lookup includes further information about the variants of interest by linking to external resources including Ensembl (Ruffier *et al.*, 2017), NCBI Genome Data Viewer (`https://ncbi.nlm.nih.gov/genome/gdv/`), UCSC Genome Browser (Kent *et al.*, 2002) or the variant browsers of the gnomAD (Karczewski *et al.*, 2020) and TopMed/BRAVO (Taliun *et al.*, 2021) projects. The same access is available directly from the users command line via tabix, as well as a custom API (`https://cadd.gs.washington.edu/api`) that enables computer programs to retrieve variant scores. Additional files from model training are available at `https://cadd.gs.washington.edu/training` and are intended for users that want to replicate how CADD models are created or build their own variant classifier upon CADD.

For each CADD version since CADD v1.3, a bigWig (Kent *et al.*, 2010) file of the

maximum SNV score per genomic position is provided. To visualize CADD scores in genome viewers like the UCSC genome browser, Ensembl Gene Browser, NCBI Genome Data Viewer or Integrative Genomics Viewer (IGV, Robinson *et al.*, 2011), the bigWig files can be loaded as browser tracks via a track hub (`https://krishna.gs.washington.edu/download/CADD/bigWig/CADD-browserTracks/hub.txt`, Raney *et al.*, 2014). This form of score representation enables users to screen large genomic ranges quickly (Fig. 24).

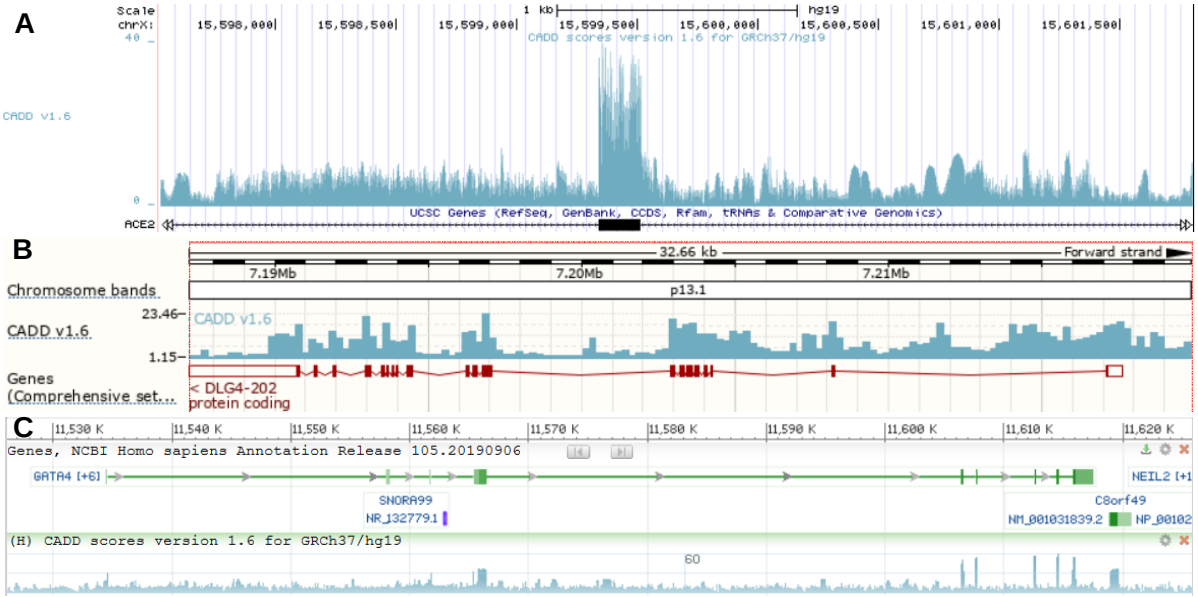

Figure 24: **CADD score track in different genome browsers.** A genome browser track hub, originally developed for the UCSC Genome Browser (**A**) displays the highest CADD SNV score for each genomic position. The tracks are available for all CADD versions and can also be displayed in Ensembl Gene Browser (**B**) and NCBI Genome Data Viewer (**C**).

In addition to the website, SNV scores are available as a plug-in for Ensembl VEP and through a number of public web resources like dbNSFP (Liu *et al.*, 2016), the Ensembl Gene Browser, ANNOVAR (Wang *et al.*, 2010), SeattleSeq (Ng *et al.*, 2009), ExAC/gnomAD, TopMed and PopViz (Zhang *et al.*, 2018). For some of these third-party sources, it can be hard to determine which version of CADD is provided, with new versions often significantly delayed. For example, this caused lifted CADD v1.3 scores of GRCh37 to be provided by third-parties for GRCh38. It is important to note that these third-party sources are not developed and officially supported by the CADD authors.

Whether for privacy reasons or because millions of variants need to be evaluated, some users want to score variants with CADD on their own system. To address these requests,

offline releases of the scoring pipeline with all associated scripts and annotations have been available for downloads for all releases since CADD v1.3. Releases starting from v1.4 are available from GitHub (`https://github.com/kircherlab/CADD-scripts`) and provide an installation script that downloads all annotations as well as optionally the prescored files. As in internal processing, all dependencies are managed via conda (`https://conda.io`).

5.7 Workflow summary

The established workflow is capable of training and applying variant scoring based on very large training data sets. In total, we have integrated more than 100 different annotations that translate almost 2,000 numeric features. Model training is optimized for linear models but can be adapted to any kind of two-class supervised machine learning algorithm like gradient boosting trees or deep neural networks. The workflow is very simple to configure, enabling the selection of features and model parameters as well as the benchmarking with different types of test sets. Users can try out many different models and optimize for various objections. Using the also provided genome-wide and user scoring pipelines, the developed models can be deployed and integrated into a web service or local cluster installation. With those capabilities, the entire workflow has been used extensively for the development of CADD models over the last years.

6 Developing new CADD models

The implementation of the new variant effect scoring workflow simplified the development of new CADD models. It made it easier to integrate new annotations and train the model using different machine learning algorithms. Most of all, however, it enables the possibility to train different models that, while similar to CADD, are independent of the previously existing model.

This chapter documents the various steps undertaken to improve CADD with the new workflow. Still using the library GraphLab, it first explores different machine learning algorithms to improve the training accuracy of CADD models. Based on the finding that there are limits to this due to the way the training data is derived and after the switch to scikit-learn, an updated CADD model is developed. The chapter concludes with a presentation of an entirely new model: CADD GRCh38-v1.4, the first variant effect model that is entirely based on the human reference genome build GRCh38.

6.1 Other machine learning models

It has been an internal discussion among the CADD authors for some time, what kind of machine learning algorithm is best for developing the CADD model. However, a deeper investigation into this area was limited by the available software and technical resources. In version 1.0, linear Support Vector Machines (SVMs) were trained to develop the CADD model. With the adaption of the GraphLab library, this changed in version 1.1 to a logistic regression classifier. The same library enabled a further investigation of more machine learning algorithms. Based on the annotations from CADD v1.3, a random parameter search of different algorithms was performed in search of better CADD models. In this comparison, models were trained using logistic regression, linear SVM, boosting trees, and random forest algorithms from the GraphLab library.

Generally, the classification accuracy of CADD models on a 1% random hold-out from the training set reaches between 55% and 60%. This relatively low accuracy is attributed to mislabeling in the training set since the proxy-negative set of simulated variants is suspected to contain a high number of neutral variants while the proxy-neutral set of human-derived variants certainly also contains a number of variants that may have negative effects. With respect to that, the results from the initial boosting trees models were astonishing, achieving a hold-out accuracy of more than 99%. On a comparison benchmark of known pathogenic and benign variants, the same models did not perform

better than random guessing. Analyzing the decision structure of the tree models, the performance jump was found to be due to implementation mistakes in two annotations: DNA shape factors and mutability index. DNA shape factors had been implemented poorly for InDels which differentiated insertions from deletions. Since human-derived variants developed from an ancestral genome version and are therefore scored by switching reference and alternative variants sequence, this enabled the classification of InDels in the training set via decision trees. As it was found later that DNA shape factors do not improve the performance for SNVs, they were removed entirely from the feature set. Similarly, the mutability index was learned by the gradient boosting tree model to very precisely distinguish SNVs between the CADD training categories. The effect can be attributed to the fact that the human-derived variants from the positive training set are a fraction of the variants used to generate the mutability index.

With DNA shape factors and mutability index excluded from the feature set, a new set of classifiers was trained. These include linear models trained via logistic regression and SVM, as well as decision tree models trained as random forest and gradient boosting trees. Both logistic regression and linear SVM models were trained for up to 40 training iterations, using a grid search over the parameters regression penalty, training iterations, and class weights. Random forests were trained with a maximum tree depths of six and up to 100 trees. Gradient boosting trees were trained with maximum depths of two, six, or seven and in total up to 100 trees in the ensemble model. It was found that the training accuracies reached on a hold-out from the training set were different depending on the machine learning algorithm. Logistic regression and SVM models generally reach an accuracy short of 60% while gradient boosting trees are more than 65% accurate on the hold-out set. This apparent advantage in prediction does not transfer to the ClinVar (release of 4th April 2017) benchmark of separating known pathogenic and benign SNV (Fig. 25), where the linear model's performance metric is better than those of the tree models.

Similar observations have been reported by Quang *et al.* (2015) and Gupta (2016). In both studies, neural networks were trained using the annotated and encoded training set from CADD version 1.0. The developed models classify the category correctly for between 66% and 68% of variants in a hold-out of the training set. While the Gupta (2016) model is not publicly available, unpublished and no benchmark data on other data set exist, the DANN model from Quang *et al.* (2015) generally has not been found to outperform later CADD models (Biggs *et al.*, 2020). In fact, the results from Quang

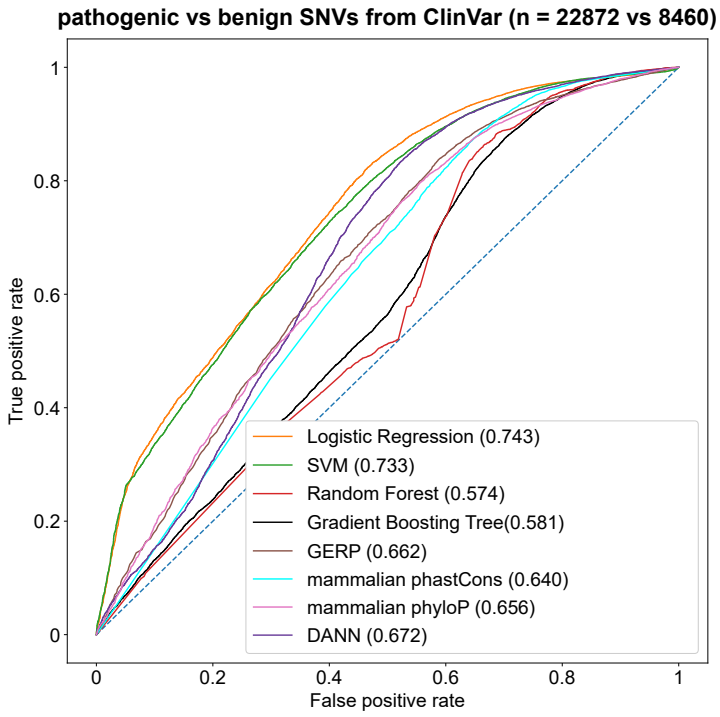

Figure 25: **Machine learning algorithms trained on CADD training data tested on SNV data.** Different machine learning algorithms can learn a model from the CADD training matrix. An auROC benchmark of those classifiers on separating known pathogenic from benign variants in the ClinVar database shows that linear models trained via logistic regression and SVM have the best performance on curated data.

et al. (2015) indicate as well that logistic regression models perform better than more complex neural networks on some benchmarks.

6.2 Estimating mislabeling in the CADD training set

It should not be a surprise that the classification rate of many models on the hold-out from the training set is worse than on the stated benchmark. As stated in the previous chapter, due to observation bias, neither the set of pathogenic nor benign variants from ClinVar are a good representation of all genomic variants. Nevertheless, pathogenicity classification is a simpler task than differentiating between variants from the set of human-derived variants versus the simulated variants, as neither of these is an accurate representation of neutral and pathogenic variants, respectively. As described in the introduction of the CADD training set, it is assumed that only between five and ten percent of all (simulated) variants are under strong purifying selection and have an

impact on gene functions. It is further assumed that a small number of human-derived variants have an impact on gene function. If we extend the number of pathogenic variants in the negative training set to 15% and assume 99% neutral variants in the human-derived variants, a theoretical best case scenario would results in an accuracy of $0.5 \times (0.15 \times 1 + 0.85 \times 0) + 0.5 \times (0.99 \times 1 + 0.01 \times 0) = 0.57$ correctly classified variants. All models achieve higher accuracies than that on the random hold-out. It is certainly possible that especially neutral variants in the two training classes come from different distributions and are therefore to some degree distinguishable on basis of the used annotations. One source, among many possibilities, could be sequence biases as observed in cancer (Alexandrov *et al.*, 2013) and DNA repair (Sabarinathan *et al.*, 2016) that are not considered in the simulation, not included in the feature set but lead to enrichment in other features.

Different neutral variant distributions do not explain why linear models like logistic regression and linear SVM are performing differently compared to decision trees and neural networks. One hypothesis that would explain this phenomenon are nonlinear interactions between features. In a linear model, a feature is given more weight by assigning a larger coefficient or less weight via a smaller coefficient. In the CADD feature set, this is extended by multiple defined feature crosses. In a decision tree or neural network, many different features may be combined nonlinearly. If these (hypothetical) interactions were specific to the training data and not related to the functional activity of a genome segment, they would explain a better training accuracy while adding no benefit to the pathogenicity predictions.

Another hypothesis is that the relative scaling of the features in a nonlinear model overfits the data. The annotations that contribute the most to the total variance of the final CADD scores are conservation scores like GERP, phastCons, and phyloP. Among these, phastCons values calculated from the alignment of ten primate genomes have had repeatedly the largest coefficient in the final models. The same primate phastCons score is the first feature that is selected by the gradient boosting tree algorithm to classify variants. PhastCons is a numeric feature defined in a range from 0 to 1. Where a linear model can integrate the gradual increase in conservation level via a single coefficient, a decision tree has to perform multiple threshold-based steps to classify the continuous score in multiple bins. While this means that the gradient boosting trees may need to learn many more parameters to achieve continuous grading of the input score, it enables that the feature can be integrated nonlinear and therefore non-monotonous. As

the relative number of negative training variants per primate phastCons conservation bin is not increasing consistently over the entire score range (Fig. 45), a nonlinear interpretation may enable the model to better evaluate the score. Furthermore, it was found that the first decision tree learned by a gradient boosting tree model evaluates primate phastCons multiple times over the decision structure from the root node to the tree leaves. For the gradient boosting trees with a smaller maximum tree size of two (a tree with one root split and two branch splits for a total of four leave nodes), multiple consecutively learned trees pick the same feature for the root split. To test the hypothesis of whether suited feature transformations may improve the learning rate, a non-uniform, rank-based transformation of the three phastCons scores in CADD was performed. For each possible value in the score range from minimum to maximum, the transformed score was calculated as the fraction of variants from the positive set among all variants from the CADD training set that have a greater conservation score than that value. Since this transformation would not be monotone, for values where a smaller value would be assigned to a higher score, that value is also assigned the higher score. Using this transformed phastCons score as a feature, a new logistic regression model reached higher accuracy on the hold-out of the training set of 62% but did not, in accordance with the results of the boosting-tree models, achieve better performance on predicting pathogenic from benign ClinVar variants.

6.3 CADD GRCh37-v1.4

At this point in the development history of CADD, the classifier was trained from over 30 million variants that were annotated from 29 main sources and more than 50 different files plus Ensembl VEP. After Ensembl database release 76, VEP (McLaren *et al.*, 2016), a dependency that annotates gene, transcript, and protein information to each variant, deprecated the previously used output format. At the same time, the system of custom annotation scripts had grown harder and harder to maintain which limited the extension to new annotations. The underlying software workflow was therefore refactored, as described in more detail in chapter 4.1. Included is a change in the Ensembl VEP output format was adapted, which enabled the integration of Ensembl database release 92. The annotation model was developed into a flexible configuration system that enables the addition or removal of annotations, feature transformations, and model parameter tests via configuration files. The entire workflow was summarized as Snakemake pipelines that, in addition to annotating the training set and training the model, integrate automated

Table 5: Annotation changes in CADD GRCh37-v1.4

removed	**DNA shape factor**	(Zhou *et al.*, 2013), lead to overfitting, undefined for InDel
	Mutability index	(Michaelson *et al.*, 2012), uses reference sequence (and derived alleles) for calculation
updated	**Protein domain annotation**	based on VEP, data format change
	Amino acid exchange	masked exchanges that cannot appear in SNV
new	**Mutation density**	distance to next variant, number of variants in 100, 1k and 10k bp window, based on gnomAD release 2.0.1 (Karczewski *et al.*, 2020)
	dbscSNV	(Liu *et al.*, 2016), splice prediction based on two machine learning models

model validation and benchmarking on separately configured test data sets.

Enabled by the workflow changes and based on several test results, multiple annotations were added, updated, or removed (Tab. 5). Two annotations that had just been introduced into CADD in version 1.1 were excluded from the feature set. The mutability index was replaced by a mutation density annotation based on SNV counts from the gnomAD database. Previous versions of CADD had further added several annotations to the output files, including allele frequencies in the 1000 genomes data set or the pathogenicity score fitCons (Gulko *et al.*, 2015), that were never used by any of the models. While originally provided for convenience, these annotations had created confusion among CADD users and were therefore dropped from the annotation files. The machine learning implementation used in the pipeline was switched from the by then deprecated GraphLab library to the free and open-source python library scikit-learn (Pedregosa *et al.*, 2011).

CADD GRCh37-v1.4 is the first CADD version that is not named only with a version number (e.g. v1.2) but also the reference genome build. All previous versions of CADD omit the genome build and are exclusive to GRCh37. The introduction of the genome

build in the version string became necessary to distinguish CADD GRCh37-v1.4 from CADD GRCh38-v1.4, which is introduced in the next chapter.

6.4 Variant effect scoring on another reference genome

In 2013, the Genome Reference Consortium (GRC, `https://www.ncbi.nlm.nih.gov/grc`) established a new reference for the human genome. Genome Reference Consortium Human Build 38 (GRCh38), also named hg38 by UCSC, is a de novo genome assembly that was generated by sequencing clone-based biological replicates. In comparison to the previous genome build GRCh37, GRCh38 corrects numerous assembly issues, integrates alternative haplotypes for complex genomic loci, and extends the genome sequence to the centromeric regions of the chromosomes (Church *et al.*, 2015). The total sequence length is increased by about 150 million bp. GRCh38 increases the robustness of genetic analyzes by minimizing artifact results and enables new biological insights (Schneider *et al.*, 2017). In the latest (as of March 2021) minor patch release GRCh38.p13, the genome reference sequence includes a total of 3,099,706,404 bp of DNA in 998 contigs.

Establishing a variant prioritization framework for a new reference genome is a major project. After all, any application that uses the, in name, same score on a different genome reference assumes that the score is mostly equivalent to the previous version. Otherwise, it would be a completely new score. For that, the software framework can remain the same as long as it is not hard coded but parameterized and configurable. The main challenge is that the training set and all features have to be replicated for the new reference genome. For annotations where this is not possible, equivalent data sets have to be found and validated. To ensure continuity, it is necessary to perform checks like correlation analyses and test results. While the new genome reference may later enable extensions that were missing or impossible in the old version, the first application on a new reference genome should not divert too much from the previous version.

In principle, it is possible to transfer variant effect scores to other genome builds with genome liftover. Liftover from GRCh37 to GRCh38 has been done for many variant prioritization scores by projects like dbNSFP (Liu *et al.*, 2016). The simplicity of liftover enables scores and annotations to continue in the new genome coordinates. However, it does not solve any of the problems why the new genome reference was established in the first place. All genome segments added or fixed in the new references will be missing from a lifted score. It also creates ambiguity for variants when the genome segment with the variant's location in the primary reference is split into multiple segments in the new

one or vice versa.

GRCh38 annotations

We preferred a solution where variants and annotations are generated directly for GRCh38. In principle, this does not require a lot of ingenuity as raw genomic data should be possible to be processed again by, instead of mapping reads to GRCh37, mapping them to GRCh38, and from then on running the entire pipeline again. Unfortunately, most genome data sets are not generated by entirely reproducible workflows. Some of this is due to hard coding when software is developed with a single data set in mind and therefore is optimized for that data set and inflexible to adapt to other data. More time-consuming however is insufficient and imprecise documentation of how software can be installed on a new computer system and how previous analyses have been run. While rare, another huge problem are software bugs, previously unknown errors in how a program executes. Given a new data set that is handled differently or has more examples than anticipated, it may fail to generate a result. Especially problematic are partially corrupted outputs, as the person doing the analysis may, if it is not the developer of the software, not be able to spot that an error occurred. Finally, the integration of annotations with small changes in the data format is always at risk of producing incorrect annotations.

The best example for what can go wrong with the adaption of an annotation for a new reference genome is the generation of the GERP (Davydov *et al.*, 2010) conservation score for GRCh38. The GERP annotation for GRCh37 had been downloaded from the GERP website `http://mendel.stanford.edu/SidowLab/downloads/gerp/` in a base-wise format. The website also provided the source code to generate the conservation score but did not specify any further information about the genome alignments used. In correspondence with some of the original authors, the GERP source code had been built and run on the BIH computing cluster using the 100 species vertebrate alignment from `http://hgdownload.cse.ucsc.edu/goldenpath/hg38/multiz100way/`. Based on these alignments, base-wise GERP-N and GERP-S scores were generated separately for each chromosome. In a second step, conserved GERP elements were then calculated from these scores. The two generated files, one for GERP-N and GERP-S and one for GERP elements, were in the same format as those downloaded for GRCh37: GERP-N and GERP-S are stored per genome position, with four tab-separated columns CHROMOSOME, POSITION, GERP-N, and GERP-S value. The GERP elements are stored in a bedGraph-like format, with five tab-separated columns CHROMOSOME, START,

END, GERP-RS score of the element and the associated P-VALUE. The first of these two files were found to be unnecessarily large as subsequent positions often have the same conservation score. It was therefore converted into a bedGraph-like format as well, with the five tab-separated columns CHROMOSOME, START, END, GERP-N, and GERP-S. Unfortunately, the annotation with this new file format was not tested thoroughly enough. Normal annotation of SNVs and InDels worked as intended. However, genome-wide annotation of SNVs for whole-genome scoring of a CADD model is not run via normal tabix file accesses but by streaming all annotation files at the same time. This did not work as intended for the new GERP file which meant that the prescored SNVs were in some cases annotated with the wrong GERP score. The CADD Phred-score calculation was then based on these scores. Since InDels and SNVs scored at run time were annotated correctly, this lead to a situation where most SNVs were associated with two CADD scores. While the mistake was easily corrected in the source code, it unfortunately affected the prescored files. Since edits to those files may produce version incompatibilities for the CADD users, it was decided to publish corrected scores with a new CADD version. The GERP annotation error, as well as another annotation error with the TFBS annotation, were limited to the new reference. With the fixed annotations, a new model was trained using the same pipeline as before. The new CADD model and the associated score were published as CADD GRCh38-v1.5.

Unfortunately, this was not the end of the problems with GERP. As stated before, the GERP conservation score is calculated for entire chromosomes. This means that the GERP software reads multiple-alignment files for a chromosome and then calculates the conservation for each position in those alignments. This calculation is, among other things, based on the GC content of the chromosome. The GC content of a chromosome is calculated by counting all bases in all genomes in the alignment. Base counts were implemented in the GERP source codes as signed 32-bit integers. With the new 100 species alignment, the variable that counts the total nucleotide number exceeded the maximum value for signed 32-bit integers, $2^{31} - 1 = 2,147,483,647$. In a process known as integer overflow, counting continued from the smallest number that is possible to store in such a variable: -2^{31}. All following calculations were affected by this mistake and produced an incorrect score. To add to the confusion, the problem did not affect GERP scores for all chromosomes but only those where the total alignment length was greater than the 32-bit integer limit. Once the error had been discovered, it was simple to fix the issue by replacing the affected variables with 64-bit integers. From there, it

was necessary to generate new GERP scores, GERP conserved elements, and ultimately a new CADD version. As with the previous error, this issue was reported by a CADD user to the development team after the release of CADD GRCh38-v1.5. The fix was integrated into the next release CADD GRCh38-v1.6.

In the end, not all GRCh38 annotations were generated newly from raw data (Tab. 6). Some, including the microRNA binding site predictions mirSVR and targetScan, transcription factor motifs, and dbscSNV where raw data was not available, had to be lifted from GRCh37. The bStatistic conservation score (McVicker *et al.*, 2009) is even older and was lifted from hg18/NCBI Build 36.1. Nevertheless, most other annotations were either available for GRCh38 from the original data providers or could be generated using available workflows. The training set was generated using the same process as for GRCh38 from EPO primate alignments to GRCh38. The GRCh38 set of human-derived variants contains 14 million SNV and 1.5 million InDel, about 1.3 million variants less than for GRCh37, and was matched with the same number of simulated variants.

Training a model for GRCh38

Using the GRCh38 annotations as features, it became possible to train a CADD model for GRCh38. CADD GRCh38-v1.4 was trained with the same workflow as CADD GRCh37-v1.4. Employing the L-BFGS logistic regression algorithm implemented in scikit-learn with the L_2-regression parameter $C = 1$, the model was converged for nine iterations. After fixing errors in the GERP and ReMap annotations, another CADD release followed. Using the same workflow and parameter settings, that model was converged for thirteen training iterations and is released as CADD GRCh38-v1.5. No CADD model version 1.5 was released for GRCh37.

The CADD releases for GRCh37 are highly correlated. On 100,000 SNVs, randomly selected from the prescored files, Phred-scores of versions 1.0 to GRCh37-v1.4 have pairwise Pearson correlations of 0.85 to 0.97 (Fig. 26 bottom-left). On the same variants lifted to GRCh38, the two GRCh38 models are highly correlated as well (Pearson correlation coefficient 0.98). For comparability, the GRCh38 scores can be lifted back to GRCh37. Using this type of analysis, the CADD releases GRCh38-v1.5 and GRCh38-v1.6 have Pearson correlations of between 0.66 to 0.79 to the GRCh37 releases, with the highest correlation with the CADD GRCh37-v1.4. A lower correlation is expected as the annotations are not perfectly identical between the genome builds.

The correlation between versions is generally lower when limited to the highest $\sim 10\%$

Table 6: CADD annotations for GRCh38

Gene and transcript annotations processing as for GRCh37 via VEP, based on Ensembl database release 92

Grantham unspecific of genome reference

phastCons, phyloP, GERP score calculation based on 100 vertebrate alignment of GRCh38 from UCSC genome browser

bStatistic liftover from hg18/NCBI Build 36.1

mirSVR liftover from GRCh37

targetScan liftover from GRCh37

chromHMM adopted chromHMM 25 state model for GRCh38, limited to ENCODE/Roadmap cell types

mRNA expression, nucleosome position, histone modification, open chromatin ENCODE reference epigenome data from 10 to 14 cell types, including assays of total RNA-Seq, DNase-Seq and ChIP-Seq for 10 different histone modifications; each is included as one feature with sum across cell types and one feature with maximum across cell types; all signals are log-transformed

Segway replaced by Ensembl Regulatory Build (which is based on a Segway model, Zerbino *et al.*, 2015)

tOverlapMotifs liftover from GRCh37

TFBS replaced by data from ReMap2 (Chèneby *et al.*, 2018), for every genomic position the number of different TFs and the number of TF-cell-type hits are counted

mutationDensity, nearestMutation same processing as for GRCh37 based on BRAVO/TOPMed freeze 5

dbscSNV liftover from GRCh37 by the authors

of variants. 10,079 of the 100,000 randomly selected variants have a Phred-score greater than 10 in CADD GRCh37-v1.4. The GRCh37 CADD releases have Pearson correlation coefficients on these variants in the range of 0.60 to 0.94 (Fig. 26 top-right). GRCh38 CADD releases have a correlation of 0.97 to each other and between 0.43 and 0.81 to the GRCh37 releases. As for the entire 100,000 SNVs of this data set, CADD GRCh37-v1.4 predictions have the highest correlation with the GRCh38 predictions among all GRCh37 releases.

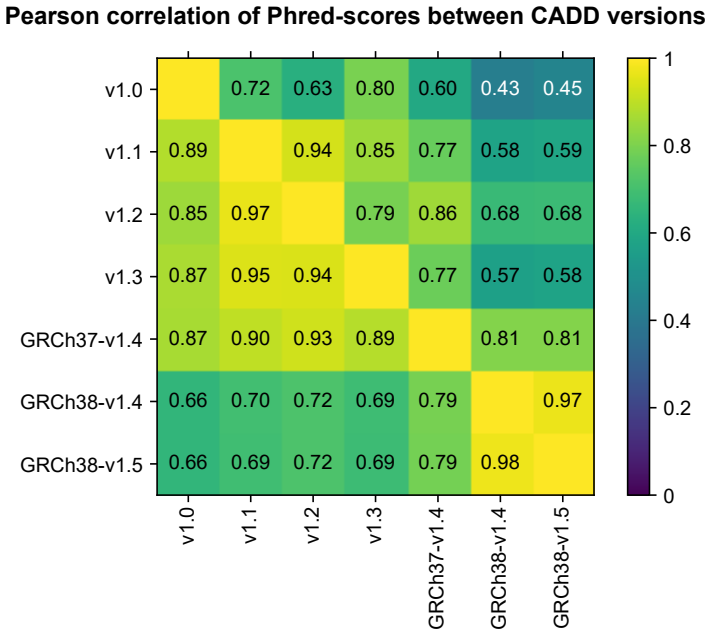

Figure 26: **Correlation between CADD versions.** Pearson correlation coefficients of Phred-scores between CADD models up to version 1.5. Correlations coefficients are calculated on Phred-scores from 100,000 randomly selected SNV throughout the genome (below the diagonal) and limited to those with Phred-scores greater than 10 in CADD GRCh37-v1.4 (n=10,079, above the diagonal)

6.5 Performance in comparison to other variant effect scores

In order to compare the performance of the different CADD releases, known pathogenic variants from ClinVar (release of July 29th, 2018) were scored and compared to frequent variants ($MAF > 0.05$) from ExAC release 1 (Exome Aggregation Consortium *et al.*, 2016). Using the auROC metric as benchmark, CADD GRCh37-v1.4 outperforms CADD v1.3 0.981 to 0.974 on all SNVs in the data set (Fig. 27A). Limited to short InDel (Fig. 27C), the results of the two versions are very similar (v1.3: 0.895, GRCh37-v1.4: 0.896).

Since the two sets of pathogenic and frequent variants come from different sources and are distributed unequally over the chromosomes, a subset of matched coding effect SNVs is selected. To perform the selection, all variants in both sets are annotated with Ensembl VEP and filtered for coding effect SNVs (non-synonymous and stop-gain variants plus all variants at canonical splice site). The numbers of variants per annotated gene are counted for each set and matched for the largest common set so that the number of variants per gene is the same in both sets. In total, 1,288 variants in 868 genes are selected per data set. Using the matched SNVs as a benchmark, GRCh37-v1.4 has an auROC of 0.901 versus 0.894 by CADD v1.3 (Fig. 27B).

Variants were compared across genome builds by lifting the initial data set from GRCh37 to GRCh38 and back from GRCh38 to GRCh37, using CrossMap (Zhao *et al.*, 2014). The repeated liftover is performed in order to remove variants from the test set that do not lift successfully between genome references. The GRCh38 models were scored with variants lifted again from GRCh37 to GRCh38 and the generated scores were lifted back from GRCh38 to GRCh37 to enable the comparison to the scores of the GRCh37 models. The GRCh38 models generally reach similar auROC scores as GRCh37-v1.4, with GRCh38-v1.5 having the best results on all SNVs (GRCh38-v1.4: 0.979, GRCh38-v1.5: 0.982) and the coding effect SNVs (GRCh38-v1.4: 0.898, GRCh38-v1.5: 0.903), while GRCh38-v1.4 predictions are slightly better on the InDel set (GRCh38-v1.4: 0.896, GRCh38-v1.5: 0.895).

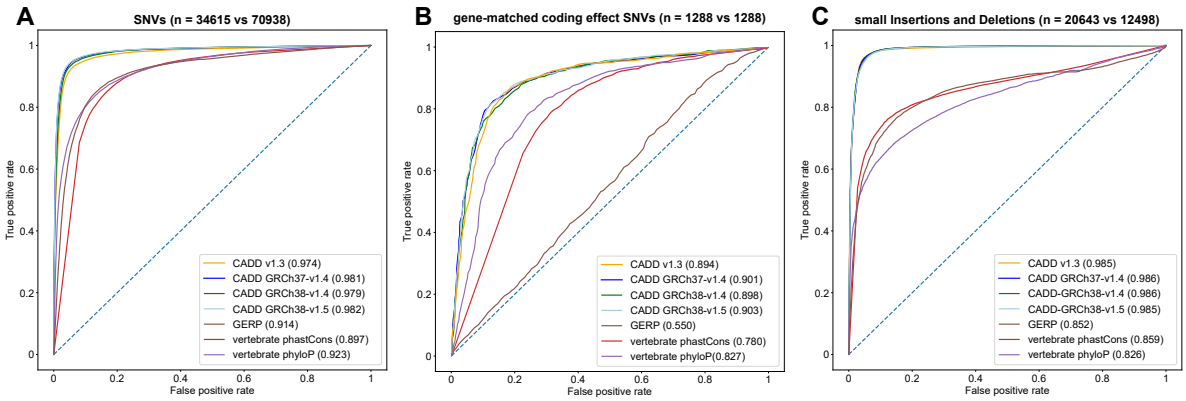

Figure 27: **CADD v1.4 improves on separating pathogenic ClinVar from frequent ExAC variants.** ROC plot of different CADD versions and some of its annotations on differentiating between pathogenic variants from ClinVar and variants from ExAC with MAF > 0.05. Both, CADD GRCh37-v1.4 and CADD GRCh38-v1.5, achieve greater auROC on all SNVs (**A**), a set of gene-matched coding effect variants (**B**) and all InDels (**C**) in the data set.

In order to compare CADD to other methods, variant effect scores and similar metrics were obtained from different sources. Eigen and Eigen-PC (Ionita-Laza *et al.*, 2016), FATHMM (Shihab *et al.*, 2014), FATHMM-XF (Rogers *et al.*, 2018), LRT (Chun & Fay, 2009), MutationAssessor (Reva *et al.*, 2011), MutationTaster (Schwarz *et al.*, 2014), PrimateAI (Sundaram *et al.*, 2018), PROVEAN (Choi *et al.*, 2012), REVEL (Ioannidis *et al.*, 2016), SiPhy (Garber *et al.*, 2009) and VEST (Carter *et al.*, 2013) were downloaded from dbNSFP (Dong *et al.*, 2015; Liu *et al.*, 2016). PolyPhen-2 (Adzhubei *et al.*, 2010) and SIFT (Ng & Henikoff, 2003) values were obtained from Ensembl VEP. DeepSEA (Zhou & Troyanskaya, 2015) scores are obtained by uploading variant files to the DeepSEA web server `http://deepsea.princeton.edu` and selecting the Disease Impact Score prediction from the results. CDTS (di Iulio *et al.*, 2018), DANN (Quang *et al.*, 2015), FunSeq2 (Fu *et al.*, 2014), LINSIGHT (Huang *et al.*, 2017) and REMM (Smedley *et al.*, 2016) are available for download as genome-wide, precomputed files. CDTS scores were downloaded from `http://www.hli-opendata.com/noncoding/Pipeline/CDTS_diff_perc_coordsorted_gnomAD_N15496_hg19.bed.gz`. It has to be noted that due to the commercial interest of the involved parties, the CDTS scores available for public download are not identical to the proprietary scores from the original publication. DANN scores were downloaded from `https://cbcl.ics.uci.edu/public_data/DANN/data/DANN_whole_genome_SNVs.tsv.bgz`. FunSeq2 scores were downloaded from `http://org.gersteinlab.funseq.s3-website-us-east-1.amazonaws.com/funseq2.1.2/hg19_NCscore_funseq216.tsv.bgz`. LINSIGHT scores (version 1.1) were downloaded from `http://compgen.cshl.edu/%7Eyihuang/tracks/LINSIGHT.bw` in bigWig format and converted to bedGraph format. REMM scores were downloaded from `https://zenodo.org/record/1197579/files/ReMM.v0.3.1.tsv.gz`. The variants of interest were selected from the genome-wide files using tabix (Li, 2011). Variants missing the score for a method are imputed with the median score of all variants for that method.

While CADD is designed for all variants genome-wide, most variant scoring methods are specialized to certain types of variants like coding, splicing, or non-coding variants. Accordingly, this method comparison separately evaluates coding and non-coding SNV predictions. No comparison was performed for InDels as most methods other than CADD only evaluate SNVs.

For the comparison of coding variant scores, the previously described benchmark of gene matched coding effect SNVs from ClinVar pathogenic in comparison to frequent

variants from ExAC is used (Fig. 28A). Compared to other variant pathogenicity scores, CADD releases GRCh37-v1.4 and GRCh38-v1.5 perform very well. Better performance than CADD is observed for VEST (auROC: 0.949) and REVEL (auROC: 0.927) whose model training sets overlaps with the benchmark data.

Large, comprehensive non-coding data sets are rare. The benchmark data set of variants from non-coding regions is therefore also a subset of the comparison of pathogenic ClinVar SNVs versus frequent SNVs from ExAC (Fig. 28B). The selection of non-coding variants in the two sets is based on the assignment from LINSIGHT for GRCh37. Both GRCh37-v1.4 and GRCh38-v1.5 have an auROC of greater than 0.99 on the non-coding data set while all other methods have lower auROC values.

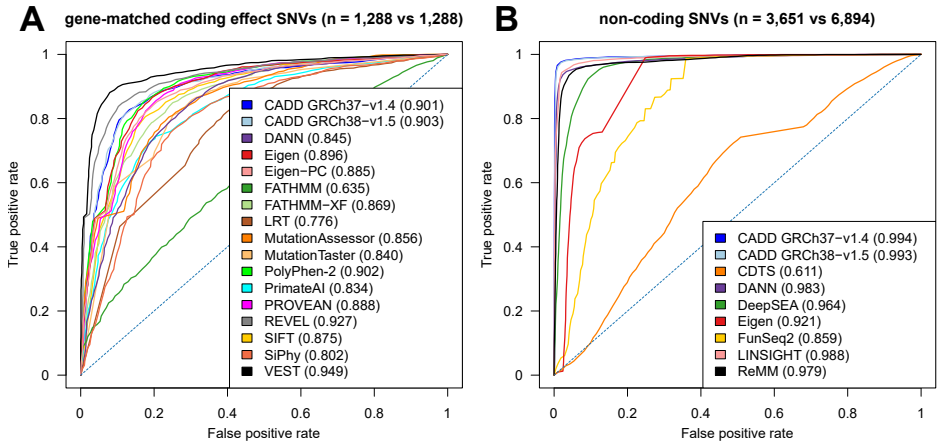

Figure 28: **Comparison of pathogenicity scores on differentiating pathogenic ClinVar and frequent ExAC SNV.** Different methods are compared on coding (**A**) and non-coding (**B**) variants. For the coding variant prediction, the set of 1,288 gene matched coding effect variants from the previous figure was used. Non-coding variants are selected as well among pathogenic ClinVar and frequent ExAC variants, with the non-coding assignment based on LINSIGHT.

It should be noted that all of the tested scores have been evaluated on ClinVar variants before. Some methods (Zhou & Troyanskaya, 2015; di Iulio *et al.*, 2018), have been specially designed to fill gaps in current predictions that are heavily based on species alignments and genome conservation and therefore do not use any conservation-based annotation for variant prediction. Studies have shown that recent evolution has altered non-coding regions of the genome which are therefore often not conserved between species (Ponting, 2017). Conservation alone may therefore be a poor predictor for non-coding variant pathogenicity. As described in the beginning, variants in ClinVar are very likely

to be under strong observations bias, so pathogenic variants in the benchmark are skewed for highly conserved variants. Other large data set are therefore necessary to better assess the different methods.

7 Improving splicing prediction in a genome-wide model

Notable among the annotations added in CADD v1.4 is dbscSNV (Liu *et al.*, 2016). dbscSNV uses two machine learning models, one AdaBoost and one random forest classifier, to predict the consequence of SNVs in the consensus region of splice sites. The splice consensus region is defined as `-3` to `+8` bp from the splice junction at donor splice sites and `-12` to `+2` bp from the junction at acceptor splice sites. All possible SNVs close to known splice-junctions have been scored with the two models genome-wide by the dbscSNV authors (version 1.1). The scores lifted to GRCh38 are available within the same files. With version 1.4, both scores were added as numeric features without transformation into CADD. A binary indicator variable was added that encodes whether dbscSNV is defined for a position. With the new annotations and on the benchmark of differentiating between known pathogenic variants from ClinVar and frequent variants from ExAC, CADD GRCh37-v1.4 has an increased performance compared to CADD v1.3 on prioritizing variants located within 20 bp of known splice junctions (Fig. 29A). All intronic variants with that 20 bp window of a splice junction, as well as all synonymous variants in the exons, are annotated as splice site variants by Ensembl VEP. For those variants, the difference in auROC is particularly large (Fig. 29B), but limited to variants where the dbscSNV annotations are defined (Fig. 29C). For splice site SNVs where dbscSNV is not defined (Fig. 29C), only a small change in auROC is observed. No improvements are observed for InDels, where dbscSNV scores are not defined.

7.1 A genome-wide data set of splicing variant effects

The set of variants used for training the dbscSNV machine learning models are partially taken from the set of pathogenic ClinVar variants used in this analysis. While the authors of dbscSNV have shown stable performance on a hold-out set, this meant that further data was needed to verify the increase in performance was not due to overfitting.

Additional validation became possible with the publication of MFASS. Multiplexed Functional Assay of Splicing using Sort-seq (MFASS, Fig. 30) is a MAVE that measures the effect of thousands of SNVs in human exons and the surrounding intronic sequences on RNA splicing (Cheung *et al.*, 2019). For each tested variant, an exon and 40 bp of intronic sequence to each side are inserted in a DNA sequence that encodes two GFP exons. The entire sequence is placed in a gene located on a plasmid that is brought into human cells. As the MFASS reporter gene is transcribed, the tested exon is either

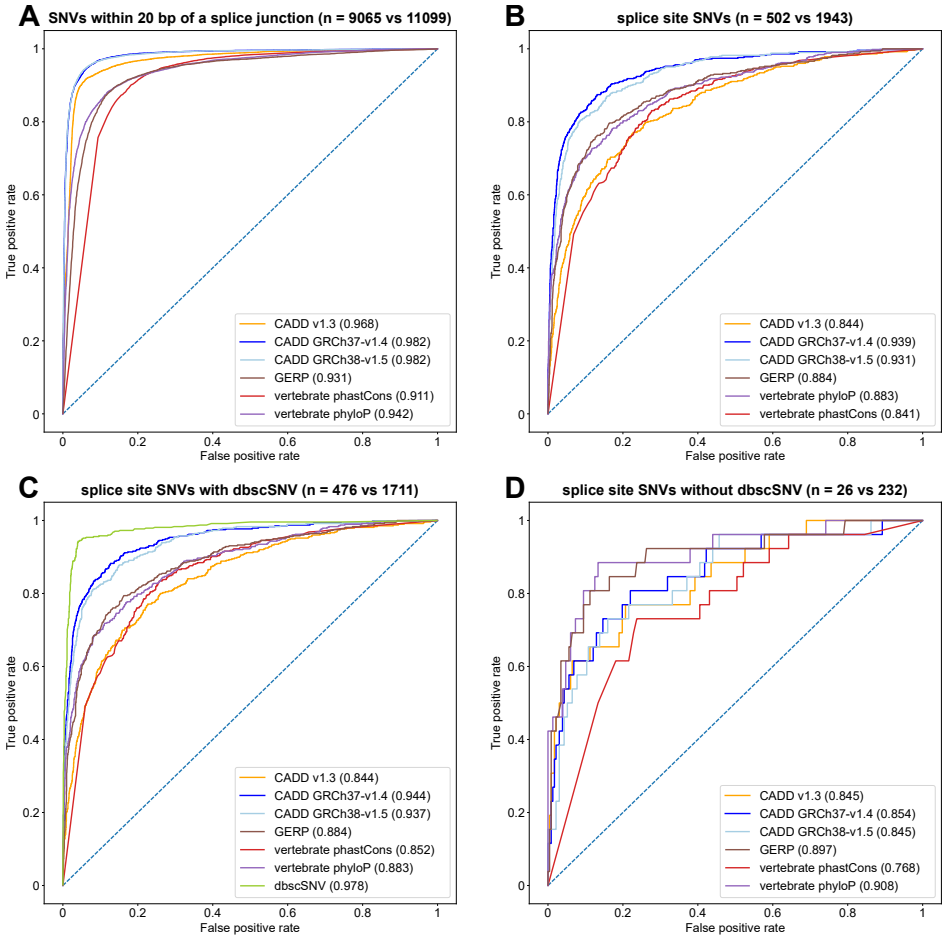

Figure 29: **dbscSNV improves the prediction of CADD v1.4 on splicing variants.** As shown in Fig. 27A, more recent versions of CADD test better than CADD v1.3 on differentiating between known pathogenic variants from ClinVar and frequent variants from ExAC. A similar increase in auROC is seen when limiting both data sets to SNVs within 20 bp of a known splice junction (**A**), especially when the test is restricted to intronic and synonymous variants (**B**). The difference may be explained by the addition of dbscSNV as a feature in CADD since the auROC on SNVs with a defined dbscSNV score (**C**) improves from 0.844 (CADD v1.3) to 0.944 (CADD GRCh37-v1.4). With these results, GRCh37-v1.4 has a higher auROC on these variants than the conservation scores GERP, phastCons, and phyloP but worse than dbscSNV itself. In C, only splice site SNVs where the random forest dbscSNV is defined are shown. For splice site SNVs where dbscSNV is not defined (**D**), the auROC increases much less from 0.845 (CADD v1.3) to 0.854 (CADD GRCh37-v1.4), and is within the range of the conservation scores that are used as features in CADD.

spliced into the final mRNA or spliced out. Total expression of the gene is the absolute number of all transcripts of that gene found via RNA-Seq. From that, the amount of splicing is calculated via the metric percent-spliced-in (psi). The psi of a genomic exon is defined as the relative fraction of total expression that this one exon is included in (Katz *et al.*, 2010). Hence, psi is always a value between 0 and 1. The change in psi (Δpsi) of a variant is determined as the absolute difference between psi values of the exon with the reference sequence and the exon with the alternative sequence. Variants with $\Delta psi > 0.5$ are identified as splice disrupting variants (sdv) and contrasted to all not splice disrupting variants (no-sdv).

All variants in the MFASS data set were originally selected from ExAC and are located in short exons or within 40 bp of the nearby intronic sequence. With more than 27,000 SNVs from over 2,000 human genes evaluated, the data set is unprecedented and exceeds previous studies of human splicing variants (Rosenberg *et al.*, 2015; Soemedi *et al.*, 2017). In total, 1,050 sdv of exonic and intronic origin were determined. In the following, splice effect prediction methods are benchmarked on differentiating between sdv and no-sdv.

The MFASS data set was downloaded from GitHub (`https://github.com/KosuriLab/MFASS/`). In total, the data set consists of 27,733 human SNVs in 2,196 exons that are positioned on the human reference genome build GRCh37. For each SNV, the data set contains the psi value that was measured using MFASS for the reference and the alternative sequence of the variant. All psi measurements are in natural scale and without any transformations. Δpsi is defined as the absolute change between the two psi measurements. The MFASS study classifies sdv based on $\Delta psi > 0.5$. All other SNVs are defined as no-sdv. For the full data set, this split results in 1,050 sdv and 26,683 no-sdv. Due to the imbalance between sdv and no-sdv that is not due to selection bias, different splicing prediction methods are compared in a PRC. As in all performance comparisons, the number of variants in every single benchmark is reduced because of variants that cannot be scored by all plotted methods, all of which are omitted.

Using the data set as a benchmark, the benefits of adding dbscSNV as a feature in CADD were analyzed. As mentioned before, dbscSNV is defined only for a short window of at maximum 12 bp around a known splice site. Furthermore, the dbscSNV models and annotations have not been published but only released as prescored variant files. Unfortunately, this means that not all canonical splice sites, as annotated by Ensembl VEP, are included in those prescored files, either because the canonical splice site was only found after the publication of dbscSNV or because it was omitted for unspecified

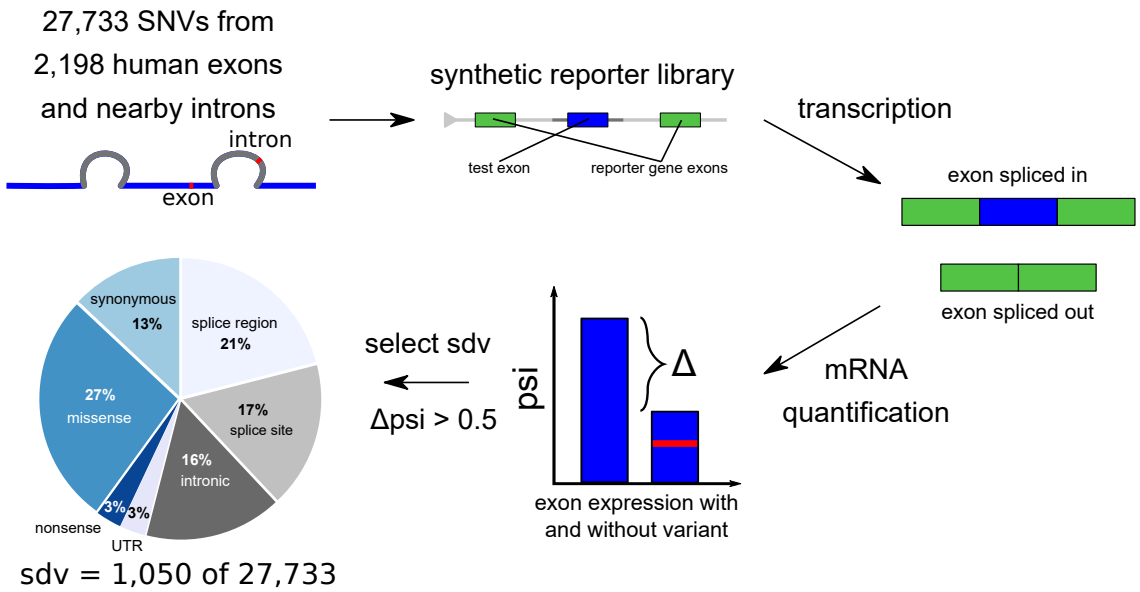

Figure 30: **MFASS reporter assay.** Multiplexed Functional Assay of Splicing using Sort-seq (MFASS) is a highly parallel reporter assay that was used to test splicing effects of 27,733 genomic variants. The variant effects are studied via a synthetic reporter library that inserts a short genome sequence of exonic and intronic sequences in a reporter gene. Transcribed mRNA of the reporter gene either integrates the exon or not, depending on if it is splice in or spliced out. Variants that cause a change in exon integration, measured in percent-spliced-in (psi), of more than 0.5 are identified as splice disrupting variants (sdv).

reasons such as sequence ambiguity or regional constraint. To estimate the effect of dbscSNV, the benchmark is split in two: (I) 3,612 variants where dbscSNV is defined (Fig. 31A) and (II) the remaining variants where dbscSNV is not defined (Fig. 31B). In the second set, CADD v1.3 and CADD GRCh37-v1.4 have an almost identical auPRC of 0.049 and 0.050 respectively, while on variants where dbscSNV is defined, the auPRC increases from 0.285 to 0.335. These results are another indication that the improvements in splice effect prediction from Fig. 29 are due to the addition of dbscSNV and that CADD predictions benefit from the new feature.

Based on these findings, further *in silico* prediction methods were explored as possible CADD features. While generalized features like conservation scores have been found to be better than random when predicting variants in intronic regions (Cheung *et al.*, 2019), specialized scores show improved performance and are necessary to successfully predict splice variants residing within exonic regions. By now, a wide variety of specialized

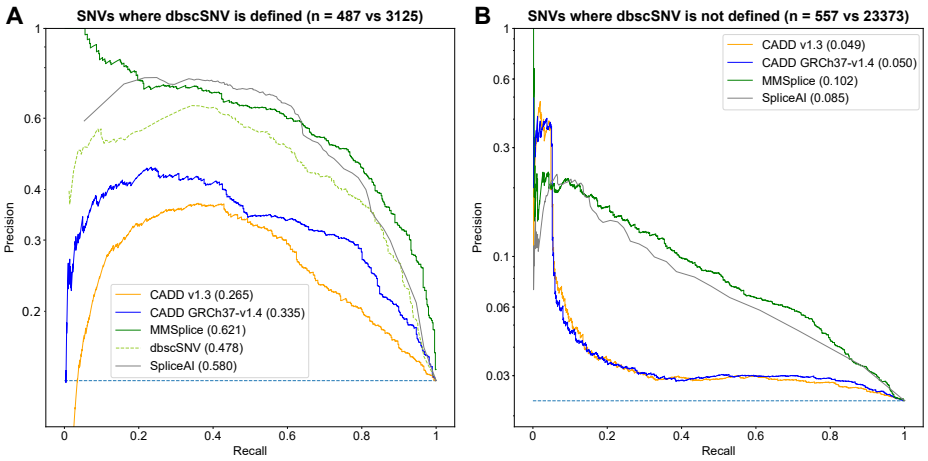

Figure 31: **CADD benefits from dbscSNV on MFASS variant prediction.** Measured in auPRC, variants where (**A**) dbscSNV is defined are better distinguished into sdv and no-sdv than (**B**) variants where dbscSNV is not defined. CADD GRCh37-v1.4 performs very similar to CADD v1.3 on **B** while it has a better auPRC on **A** where dbscSNV contributes as a feature to the model score. For variants where it is defined, dbscSNV has a higher auPRC than CADD. MMSplice and SpliceAI are defined for all variants and have higher auPRC than CADD in both settings.

methods have been developed to predict changes in RNA splicing (Jian *et al.*, 2014b). Many of these predictors have been trained via machine learning (Abramowicz & Gos, 2018), including decision tree (Mort *et al.*, 2014; Jian *et al.*, 2014a; Soemedi *et al.*, 2017; Jagadeesh *et al.*, 2019), probabilistic (Yeo & Burge, 2004) and k-mer-based (Ke *et al.*, 2011; Rosenberg *et al.*, 2015) models. The first generation of splicing scores, like MaxEntScan (Yeo & Burge, 2004), focus on the immediate neighborhood of splice junctions, as most splicing variants have been found in these regions (Abramowicz & Gos, 2018). In the last few years, more distal splicing regulatory elements have been taken into account (Mort *et al.*, 2014; Soemedi *et al.*, 2017; Jagadeesh *et al.*, 2019). Recently, deep neural networks (DNNs) achieved good results in predicting splice variants genome-wide. The idea of using neural networks for splice predictions is almost three decades old (Brunak *et al.*, 1991; Reese *et al.*, 1997). The first tool to leverage the recent progress in deep learning technology was SPANR (also SPIDEX, Xiong *et al.*, 2015), which is trained on experimentally observed exon skipping events and predicts exon inclusion percentages based on genomic features. Instead of using predefined features, two recent tools, MMSplice (Cheng *et al.*, 2019) and SpliceAI (Jaganathan *et al.*, 2019), are limited to genomic sequence as input for their prediction.

To access strengths of different methods, the MFASS data set was split into intronic (n=13,603) and exonic (n=14,130) variants based on the annotation by Cheung *et al.* (2019). Again, the number of variants in each PRC is slightly smaller as only variants were included for which all tested methods are defined.

A number of specialized methods were adopted for the MFASS benchmark. Hexamer HAL (Rosenberg *et al.*, 2015) scores were generated using HAL model scripts from Kipoi (Avsec *et al.*, 2019). HAL scores including percent-spliced-in (psi) were downloaded with the MFASS data set, originally obtained via the HAL website `https://splicing.cs.washington.edu` for exon skipping variants by the MFASS authors (Cheung *et al.*, 2019). S-CAP (Jagadeesh *et al.*, 2019) (v1.0) scores were downloaded from `https://bejerano.stanford.edu/scap/`. All eight S-CAP scores were combined into one score by taking the maximum per variant. Where specifically indicated and per S-CAP definition, variants without a precalculated score were imputed as benign (S-CAP score = 0). SPANR (Xiong *et al.*, 2015) (v1.0, noncommercial) scores were downloaded from `https://assets.deepgenomics.com/spidex_public_noncommercial_v1_0.tar`. MMSplice (Cheng *et al.*, 2019) scores were generated via the provided script release (v1.0.2) as installed from pypi. The exon-intron boundaries were provided as GTF gene annotation file downloaded from Ensembl release 95 (Aken *et al.*, 2016). The script generates model scores of the sequence with reference allele and with alternative allele for five submodels (acceptor, acceptor intron, exon, donor, and donor intron). The script also outputs the composite linear models *delta_logit_psi* and *pathogenicity* that summarize the five submodels in one metric. *delta_logit_psi* scores were used in performance comparisons. Prescored SpliceAI (Jaganathan *et al.*, 2019) v1.3 scores were downloaded from Illumina BaseSpace. For larger InDels unavailable from precomputed scores, the variant scores were computed via an adapted version of the SpliceAI scripts version 1.3 (`https://github.com/Illumina/SpliceAI/`) that is able to integrate scores from prescored files in order to enable faster scoring. In comparisons of SpliceAI with other scores, all four SpliceAI models were combined into a single score by using the maximum score for each variant.

A combined score of MMSplice and SpliceAI, MMAI, was defined for evaluation on the MFASS data set. To give equal weight to both MMSplice and SpliceAI, scores were divided by their respective standard deviation across all MFASS variants (MMSplice 0.5291, SpliceAI 0.1206) and the normalized scores added. For SpliceAI, the maximum score across all SpliceAI submodels was used while MMSplice is included via the model

combination *delta_logit_psi*. Similar to MMAI, MMAIpsi was defined by adding normalized psi as measured for the reference allele in the MFASS data set (standard deviation of 0.0622 across all MFASS variants) to MMAI. "proportion expressed across transcripts" (pext, Cummings *et al.*, 2020) was downloaded in the version of 27 of February 2019. While the file is not available from the original location anymore, the file has been archived at `https://doi.org/10.5281/zenodo.4447230`. For intronic variants, the pext value of the closest exon was annotated.

Besides dbscSNV, the splicing effect predictors methods HAL (Rosenberg *et al.*, 2015), MMSplice (Cheng *et al.*, 2019), S-CAP (Jagadeesh *et al.*, 2019), SPANR (Xiong *et al.*, 2015) and SpliceAI (Jaganathan *et al.*, 2019) were benchmarked (Fig. 32). From these methods, SpliceAI and MMSplice, both DNNs based solely on genomic sequence, showed the best overall performance (Fig. 32A) with auPRC of 0.328 (SpliceAI) and 0.361 (MMSplice). While the performance of all methods is generally better and less variable for introns (Fig. 32C) then exons (Fig. 32B), MMSplice and SpliceAI are the best performing methods on both subsets.

In the original MFASS publication, some methods such as the conservation scores PhyloP and PhastCons have very low auPRC values and appear to perform worse than if predictions had been random. This finding could not be reproduced. It was found that the authors of MFASS had inverted the values of these scores and that, when corrected, those scores perform better than random guessing on predicting splice effects. However, the predictive power of the species conservation measures for exonic variants remained limited, because most exonic variants fall in the highest conservation score bin (Fig. 32B).

Despite their similar auPRC performance on the MFASS data set, Spearman correlation between SpliceAI and MMSplice on MFASS variants is only around 0.6. While both models are trained on different data sets, the dissimilarity may be based on the different model architectures. MMSplice is a convolutional neural network that was trained on data from a large MPRA library (Rosenberg *et al.*, 2015) of random sequences and takes into account 75 bp of sequence up and downstream of a known splice junction for splice donors and splice acceptors. This is in contrast to SpliceAI that as a deep residual network takes advantage of a much larger sequence window of 10,000 bp and was trained on RNA expression data from different individuals and tissues in GTEx. Based on the speculation that as both scores were derived very differently, they may complement each other. Thus, an equally weighted linear combination of the two scores (MMAI) was

evaluated. MMAI was found to reach an auPRC of 0.416 (Fig. 32), better than either of its parents.

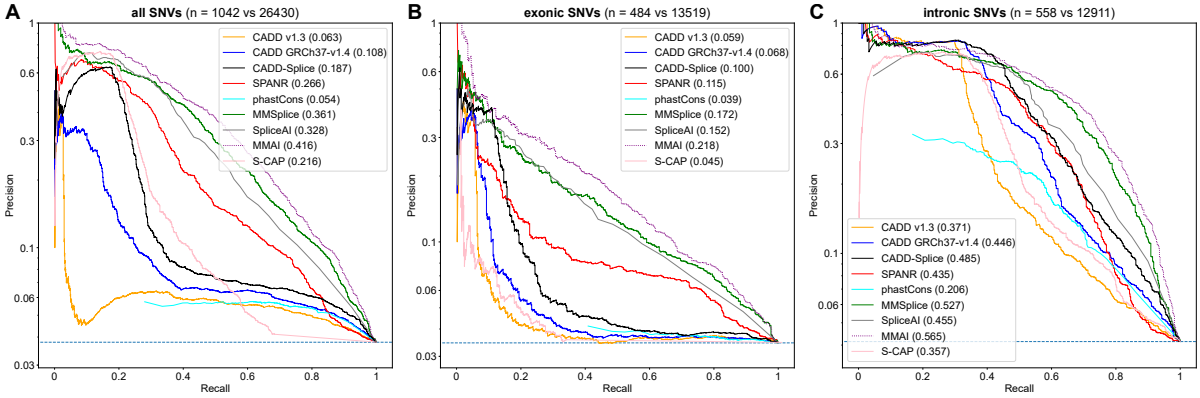

Figure 32: **Additional splicing scores improve the performance of CADD on MFASS.** PRC of various prediction methods separating sdv from no-sdv. The benchmark results are dependent on the selected variants and differ between (**A**) all variants from MFASS, limited to (**B**) exonic or (**C**) intronic variants. In all three cases, effect predictors specialized on splicing variants like MMSplice, SPANR and SpliceAI have the highest auPRC. Further improvement is available in a linear combination of the predictors MMSplice and SpliceAI (MMAI). The new CADD model CADD-Splice integrates MMSplice and SpliceAI into its extensive feature set and hence outperforms previous CADD models on the MFASS benchmark.

In Cheung *et al.* (2019), the predictor with the highest auPRC on exons is HAL (Rosenberg *et al.*, 2015, Fig. 33B). However, the sequence-hexamer-based model of HAL further integrates psi of the reference allele as an additional assay-derived source of information. Δpsi, a measure derived from psi, was used to define sdv and no-sdv variants. psi of the reference alone separates sdv from no-sdv variants (Fig. S1B, auPRC of psi 0.143, HAL with psi 0.274, HAL without psi 0.175). Any interpretation of the increased performance therefore needs to consider the underlying circularity. Adding psi in the linear combination of MMSplice and SpliceAI (MMAIpsi) results in an auPRC of 0.471 (Fig. 33A). MMAIpsi outperforms all other models on exons, where increased precision is specifically observed for high recall thresholds (Fig. 33B). Using HAL without psi does result in the same performance as MMSplice (auPRC 0.175, Fig. 33B). Since the application of HAL is by design limited to exons, MMSplice was chosen over HAL for all further analysis.

As an assay-derived measure, the psi values from the MFASS study cannot be used to predict splicing effects genome-wide. However, genome-wide availability would be

a prerequisite for including psi as an unbiased feature in variant prediction. While measures of psi can be derived for any RNA-Seq data set (Katz *et al.*, 2010; Shen *et al.*, 2012) and are predictive of specific cell-types (Park *et al.*, 2018), CADD would require an organismal summary of all cell types and developmental stages. A possible candidate metric (Ling *et al.*, 2020) became available only after this analysis had finished. Here, the proportion expressed across transcripts (pext, Cummings *et al.*, 2020) is explored, a close proxy of psi. pext is based on RNA-Seq transcript assemblies and quantifies the expression of each nucleotide in an exon in relation to the entire transcribed sequence of the gene. However, neither does pext separate sdv and no-sdv variants very well (Fig. 33A, auPRC of 0.058 versus 0.143 for psi) nor do we find any separation of splicing variants in the CADD training set based on pext value. While better equivalents may be considered, it is possible that psi values as measured in MFASS are very assay-dependent.

7.2 New splicing features in CADD

Aggregating all findings, a new CADD model was developed by integrating SpliceAI and MMSplice as features. In preparation for the integrating, the score distribution in the two training set classes was analyzed for each submodel. For SpliceAI, this included the four 10,000 bp submodels donor gain, donor loss, acceptor gain, and acceptor loss. Since the generation of genome-wide scores from sequence proved to be too computationally expensive, precomputed SpliceAI scores from Illumina BaseSpace were used. Since models require the reference nucleotide of a variant to match the human reference, variants from the proxy-benign set (human-derived variants) of the CADD training data were scored with reference and alternative alleles reversed. To adjust for this, gain and loss model scores were swapped for donor and acceptor. In both training data set and final scoring, predicted splice gains at annotated splice sites and predicted splice loss outside of annotated splice sites were set to 0 (for donor and acceptor sites). This was previously described for SpliceAI (Jaganathan *et al.*, 2019) and has been referred to as masking. Applied after the swap of gain and loss in human-derived variants, masking the SpliceAI submodels resulted in a large depletion of high scores in the human-derived class while unmasked scores (i.e. splicing loss outside of existing sites and splicing gain for already existing sites) did not show class specificity (Fig. 34).

For MMSplice, the five models acceptor, acceptorIntron, donor, donorIntron, and exon were evaluated separately. In the case of the proxy-deleterious class of simulated

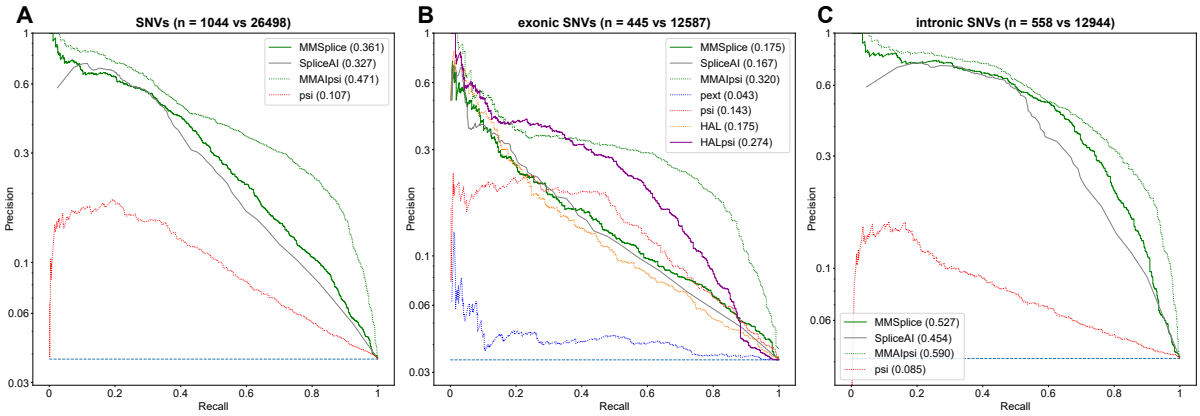

Figure 33: **percent-spliced-in is a predictor of splicing disruption.** PRC of psi, pext, HAL and other scores for (**A**) all, and limited to (**B**) exonic and (**C**) intronic variants tested with MFASS. Using reference psi value of the reporter assay as a predictor to distinguish sdv and no-sdv is far better than random guessing. In the original MFASS study, psi was used as additional input for the HAL predictor (HALpsi) which produced the highest benchmark auPRC of all predictors on exonic variants. The HAL predictor without psi does reach lower auPRC values. MMAIpsi, the combination of MMAI with psi performs better than HALpsi and is also available for intronic variants. pext, which was explored as a genome-wide alternative to psi, performs worse at distinguishing between sdv and no-sdv.

Figure 34: *Figure on next page.* Histograms of SpliceAI model scores for 10 million SNVs in the CADD training set (human-derived: blue, simulated: green). Masked variants (loss variants that influence a known splice junction (**A**) and gain variants outside of a known splice junction (**C**)) are depleted in the simulated category for variants with high scores. Other, unmasked, variants (gain at a known splice junction (**B**) or loss outside of known splice junction (**D**)) are not depleted. Plots are limited to donor gain (**A** & **B**) and acceptor loss (**C** & **D**) submodel scores. Variants not annotated by SpliceAI are omitted, with the number of variants per plot specified in the title as n = number simulated / number of human-derived variants.

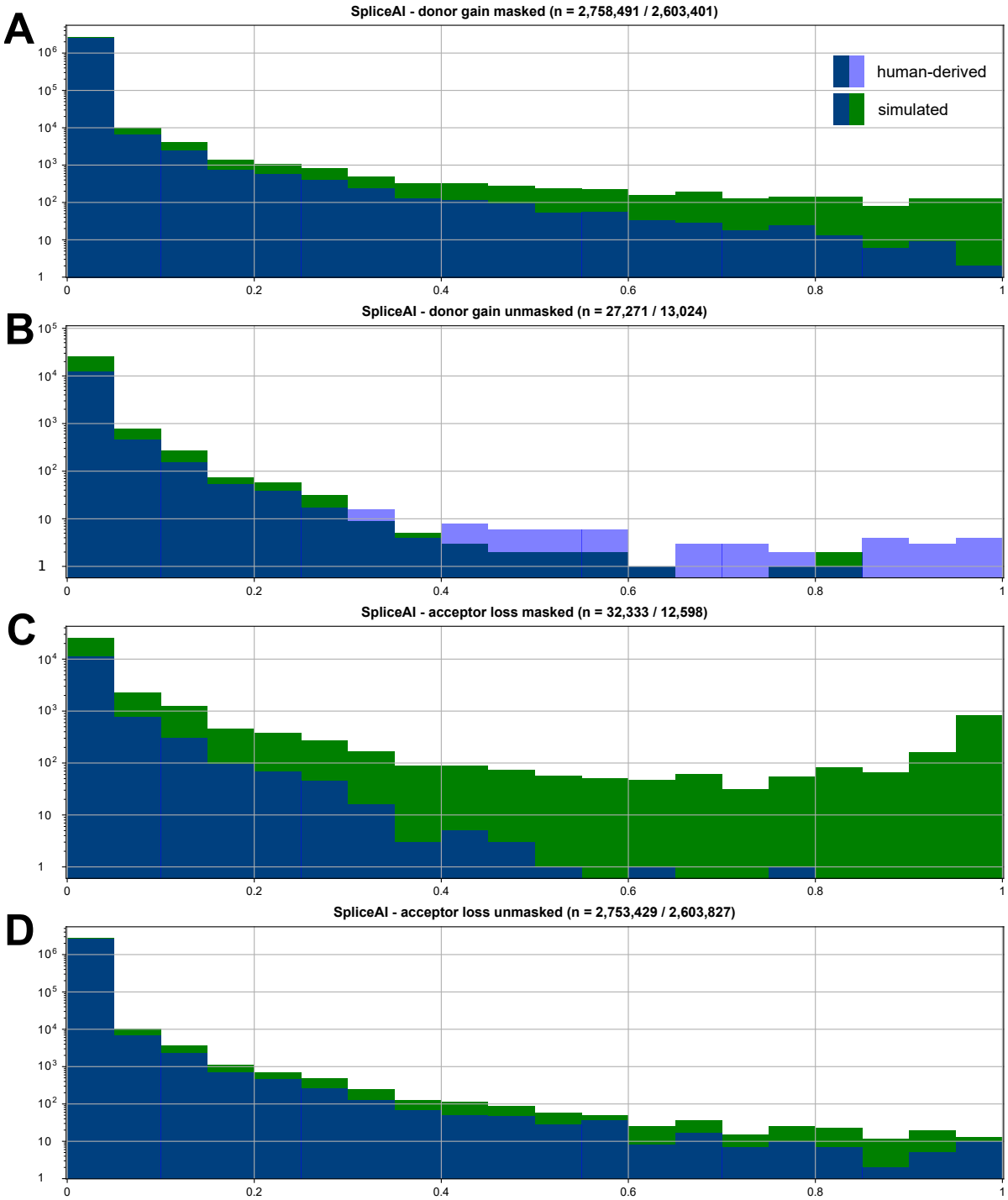

Figure 34: **Depletion for high SpliceAI model scores in the CADD training data.** *Description on previous page.*

variants as well as in scoring applications of the CADD model, the reference score was subtracted from the alternative score, as described by the authors. Since the MMSplice script provides scores only for variants where the reference matches the genome reference, human-derived variants have to be scored in reverse and the alternative score is subtracted from the reference score.

A depletion of human-derived variants was observed for negative scores, as most pronounced for the acceptor, donor, and exon models (Fig. 35). No depletion of positive scores could be observed in the human-derived variants. Instead, an enrichment of human-derived variants was observed for strong positive scores. It was concluded that positive score differences have to be set to 0 for all MMSplice submodels.

For variants annotated with multiple different consequence predictions as annotated by Ensembl VEP, both MMSplice and SpliceAI scores were limited to the consequence of the same gene. All variants not annotated by MMSplice or SpliceAI were imputed as 0. All nine MMSplice and SpliceAI sub-model annotations were further included in a feature-cross with the consequence annotation. Independent of these splicing features, the category "unknown" was removed from the categorical consequence levels as only two variants in the entire training set were annotated in this category. These variants, classified by VEP as "coding sequence variant" without further specification, were reassigned to the "synonymous" consequence category. An issue specific to CADD v1.4 was corrected, which previously made it possible that highly conserved coding variants could be scored as UTR of overlapping gene annotations. Hyperparameter optimization was performed with the same strategy as for CADD v1.4/v1.5. The final model was trained for 13 iterations with the L_2-regression parameter $C = 1$ via the scikit-learn implementation of L-BFGS logistic regression.

Figure 35: *Figure on next page.* Histograms of MMSplice model scores for 10 million SNVs from the CADD training set (human-derived: blue, simulated: green). Score distributions are different for the specialized models: acceptor (**A**), acceptorIntron (**B**), donor (**C**), donorIntron (**D**) and exon (**E**). Variants not annotated by MMSplice are omitted, with the number of annotated variants per plot specified in the title as n = number simulated / number of human-derived variants.

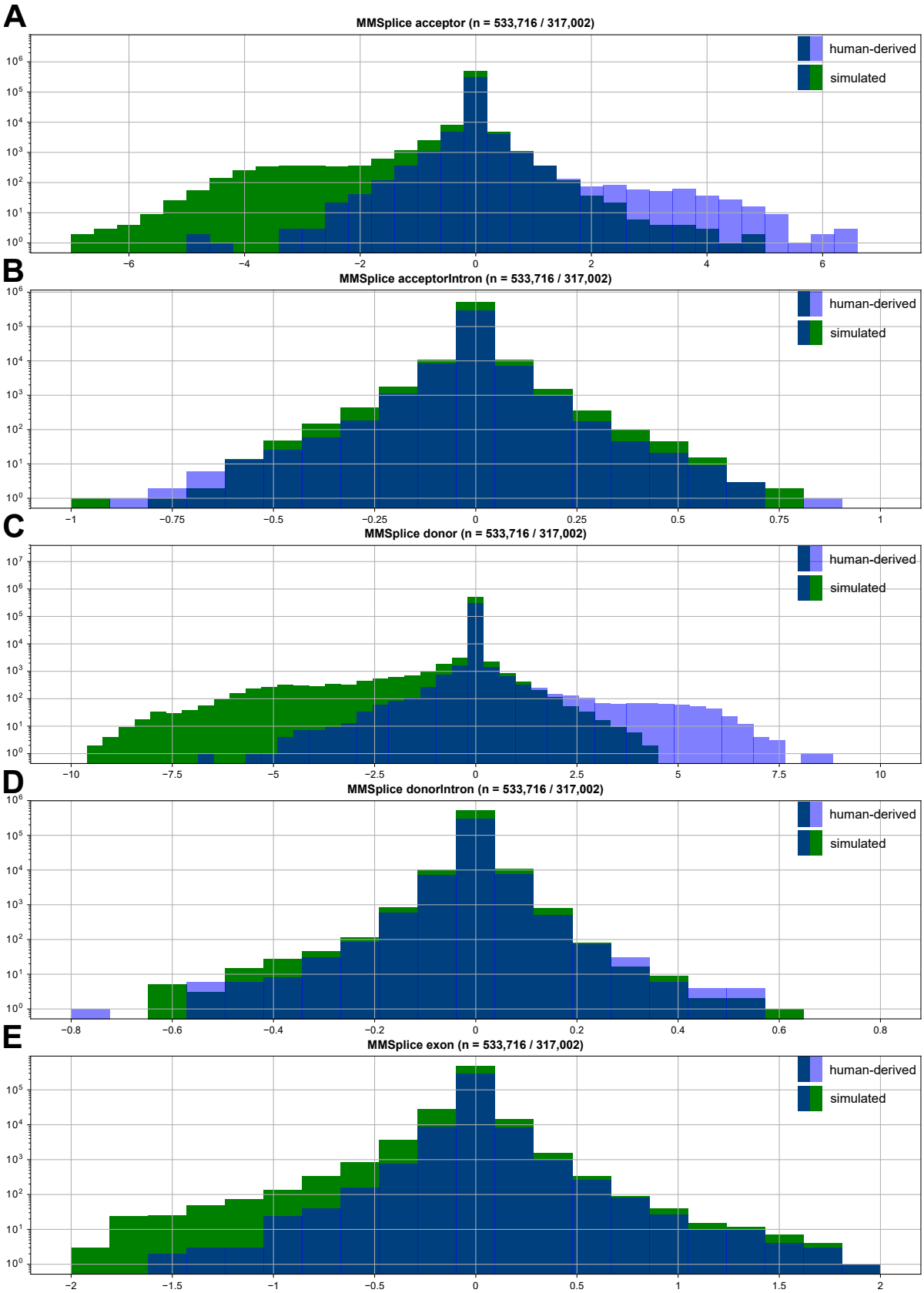

Figure 35: **Distribution of MMSplice model scores in the CADD training data.** *Description on previous page.*

7.3 Benchmarking CADD-Splice

The new model has the formal version tag CADD GRCh37-v1.6 but has been named CADD-Splice to specifically highlight the improvements on predicting splicing effects. CADD-Splice shows an increased auPRC of 0.185 on the entire MFASS data set (CADD GRCh37-v1.4: 0.108), with better performance on both exonic and intronic variants (Fig. 32). All MMSplice and SpliceAI features were learned with positive coefficients in the CADD-Splice model, which indicates that increased scores in the splice models are associated with increased deleteriousness in the combined model.

As mentioned above, in the original MFASS study (Cheung *et al.*, 2019) variants were split into categories based on the observed changes in psi: sdv have $\Delta psi > 0.5$ while no-sdv have $\Delta psi \leq 0.5$. While there is reasoning behind that threshold, other values for Δpsi may be chosen for the evaluation of in-silico predictions. A lower threshold will increase the number of variants classified as sdv, while a higher threshold will increase the number of no-sdv. In Figure 36, results are shown for thresholds 0.7, 0.3 and 0.1 (Fig. 36A-C), as well as a two-step threshold of variants with $\Delta psi > 0.5$ and variants with $\Delta psi > 0.1$ (Fig 36) that excludes uncertain variants between the two thresholds. While the general performance of each method varies slightly depending on the chosen thresholds and mostly declines with lower Δpsi thresholds, the relative ranks in terms of auPRC are the same for each setting.

The correlation of CADD-Splice with previous CADD releases was analyzed using the same type of correlation analysis described in chapter 6.4. On 100,000 genomic SNVs, CADD-Splice has a Pearson correlation of 0.99 with CADD GRCh37-v1.4 (Fig. 37). For 10,079 variants with a Phred-score greater than 10 in CADD GRCh37-v1.4, the correlation to the latter is 0.97.

Score enrichments enable the attribution of the observed differences between versions to certain variant categories. To perform the enrichment analysis, gnomAD SNV were assigned to three allele frequency groups as frequent (MAF > 0.001), rare (MAF ≤ 0.001, allele count > 1) and singleton (allele count $= 1$). Variant types were annotated using Ensembl VEP and limited to the broader consequence category used in CADD. To compare scores between versions without taking category shifts into account, CADD raw score percentiles of all variants in the same category were used as ranks. Enrichments per category were calculated by selecting for each percentile all variants of the same category with a rank larger than this percentile and dividing the number of observed variants per frequency group by the number expected from random drawing. From

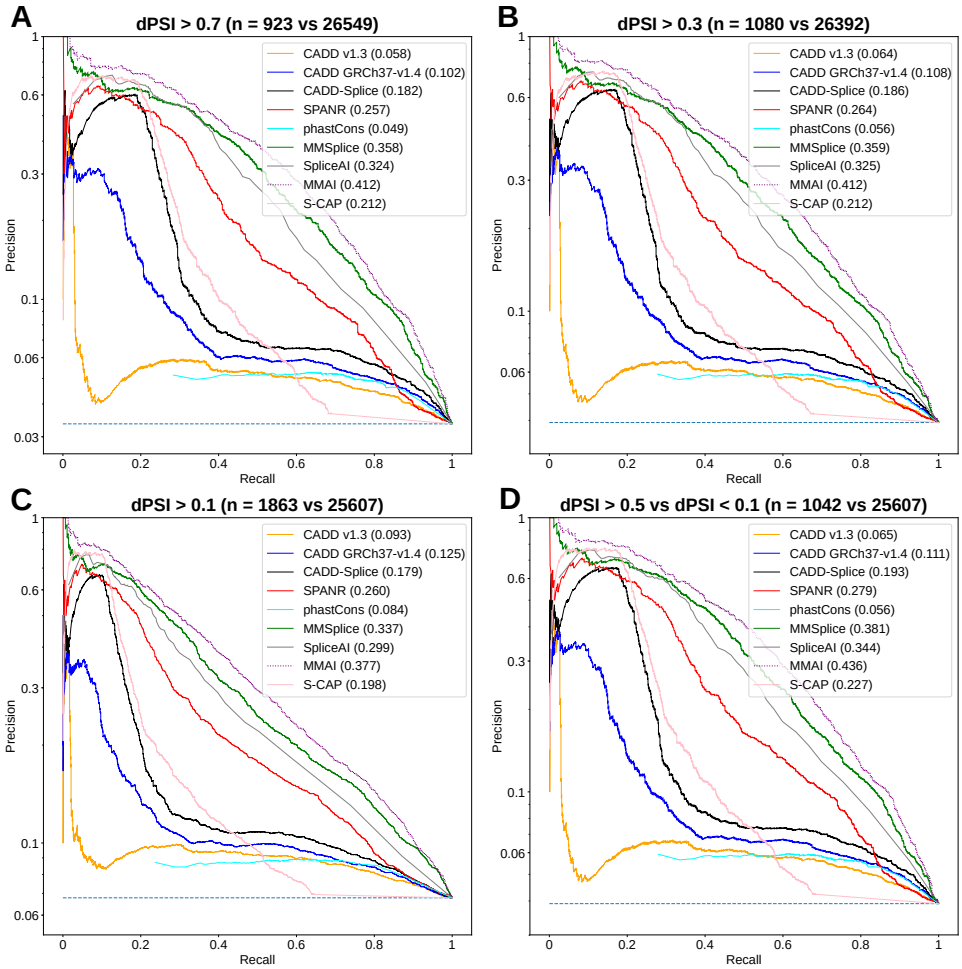

Figure 36: **MFASS auPRC rank is independent of Δpsi.** All MFASS variants are separated into sdv and no-sdv entirely based on Δpsi thresholds. Instead of the default value of 0.5, thresholds of $\Delta psi > 0.7$ (**A**), $\Delta psi > 0.3$ (**B**), $\Delta psi > 0.1$ (**C**) and a two-step threshold of $\Delta psi > 0.7$ for sdv versus $\Delta psi < 0.1$ for no-sdv (**D**) are shown.

the entire sample, 1,000 bootstrap iterations were drawn of which the 95% confidence interval is used as variance estimate.

For all categories and models, high CADD scores are depleted in the group of frequent variants and enriched in the group of singleton variants (Fig. 38). Larger differences in enrichment are observed for variants around known splice sites, as apparent from an increased depletion of high CADD scores for "frequent" variants and enrichment of gnomAD singletons in splicing related variant categories. In the splice site proximal regions within 20 bp of a known splice site, this enrichment/depletion effect increases from CADD v1.3 over v1.4 to CADD-Splice (Fig. 38A). However, for canonical splice

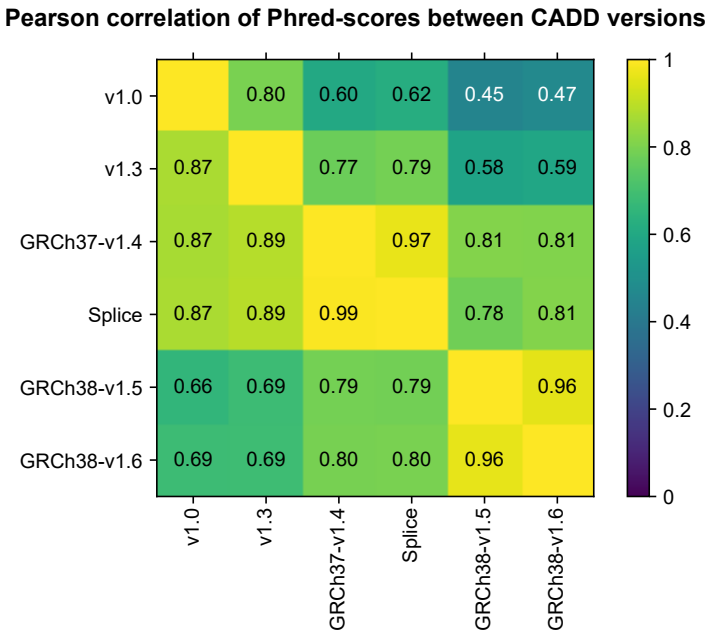

Figure 37: **CADD-Splice is highly correlated to previous CADD version.** Pearson correlation between the latest CADD models. Correlations coefficients are calculated on Phred-scores from 100,000 randomly selected SNV throughout the genome (below the diagonal) and limited to those with Phred-scores greater than 10 in CADD GRCh37-v1.4 (n=10,079, above the diagonal). CADD GRCh37-v1.4 and CADD Splice, as well as CADD GRCh38-v1.5 and GRCh38-v1.6 have correlation coefficients higher than 95%. The correlation to and between previous version is lower.

sites, changes overlap within the 95% confidence interval of CADD-Splice (Fig. 38A). For other variant categories such as intronic variants and coding mutations, score depletion of CADD GRCh37-v1.4 and CADD-Splice increases for larger percentiles compared to CADD v1.3 (Fig. 38C-E) but no significant difference is observed between the two latest releases.

As described for previous versions, performance is benchmarked on pathogenic SNVs from ClinVar in contrast to common SNVs from gnomAD that are suspected to be overwhelmingly neutral (Fig. 39). Performance is compared based on auROC. The ClinVar database (release April 20th, 2020) was downloaded from `https://ftp.ncbi.nlm.nih.gov/pub/clinvar/`. "pathogenic" SNVs were selected from the database based on the assignment of 'Variant Clinical Significance', excluding variants with multiple assignments. gnomAD (version 2.1.1) was downloaded from `https://gnomad.broadinstitute.org/`. SNVs that did not pass all filters set by the gnomAD authors were excluded. Common variants were selected by excluding all variants with a minor allele

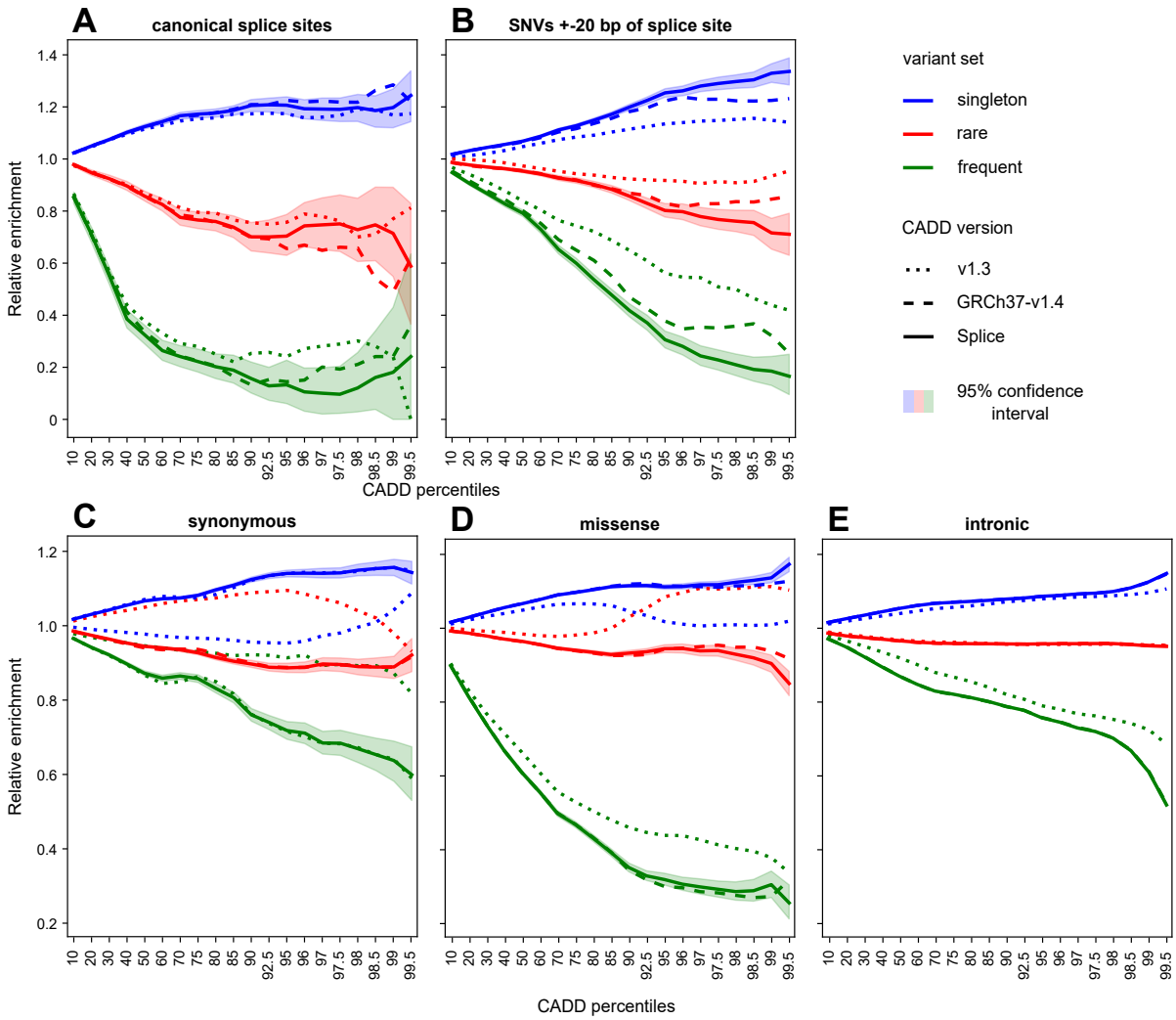

Figure 38: **Enrichment of CADD scores by variant allele frequency.** To compare enrichment, SNVs are grouped in three gnomAD frequency bins: frequent (MAF > 0.001, green), rare (allele count > 1, red) and singletons (observed exactly once in gnomAD, blue). Variants are split by impact category, with canonical splice site SNVs (**A**), variants with 20 bp of a splice site (**B**), synonymous (**C**), missense (**D**) and intronic SNVs (**E**) shown. CADD versions are separated by line-style and CADD scores are converted into percentiles of all variants per category. The color shades in the background outline the 95% confidence interval determined via bootstrapping.

frequency (MAF) less than 0.05 across all gnomAD samples. In order to later score GRCh37 variants with CADD GRCh38 models, variants were lifted to GRCh38 using CrossMap (Zhao *et al.*, 2014), excluding variants that did not lift back to the same GRCh37 coordinates. The final "benign" set contains 163,581 gnomAD SNVs and the "pathogenic" set contains 68,479 ClinVar SNVs.

On intronic (Fig.39C) and splice site SNVs (Fig.39B), CADD-Splice (auROC intronic 0.957 and auROC splice site 0.978) outperforms not only previous versions of CADD (GRCh37-v1.4: 0.879 and 0.938) but also the specialized scores MMSplice (0.886 and 0.970) and SpliceAI (0.869 and 0.959). For synonymous variants (Fig. 39D), smaller increases in auROC were observed, possibly due to a mixture of splicing-related and unrelated changes in the model. Only a few common SNVs in gnomAD are located at canonical splice sites, limiting the meaningfulness of the canonical splice site benchmark (Fig. 39E). From what can be observed, CADD-Splice does not improve the prediction on this variant class as CADD GRCh37-v1.4 has the highest auROC. The ROC curves of CADD-Splice and CADD GRCh37-v1.4 are almost identical for missense variants.

Since the ClinVar pathogenic benchmark had been used for model testing before, likely-pathogenic SNVs were compared to low-frequency variants. As in the previous benchmark, likely-pathogenic SNVs are selected based on the assigned clinical significance in ClinVar (incl. variants assigned the two terms "likely-pathogenic" and "pathogenic"). In total 27,458 SNVs are selected. The set of low-frequency variants is selected by randomly picking 300,000 SNVs with a minor allele frequency below 0.05 and an allele count above 1 in gnomAD.

The benchmark results replicate the previous results while highlighting the excellent performance of CADD on the complete variant set (Fig. 40). In addition, the benchmark allows comparison to the specialized splicing scores S-CAP and SPANR whose training set partially overlaps the ClinVar pathogenic set (Fig. 41). While SPANR does not perform better than CADD-Splice in any of the comparisons, S-CAP outperforms CADD on canonical splice site variants (Fig. 41B) and intronic SNVs (Fig. 41D). However, precomputed S-CAP scores are missing for about 9% (5,980 out of 66,608) of splicing-related variants in this test set (Fig. 41B-D). When interpreting missing variants as benign rather than excluding them from all comparisons (Fig. 41E-H), most S-CAP auROC values reduce substantially and increase only for canonical splice sites (Fig. 41F).

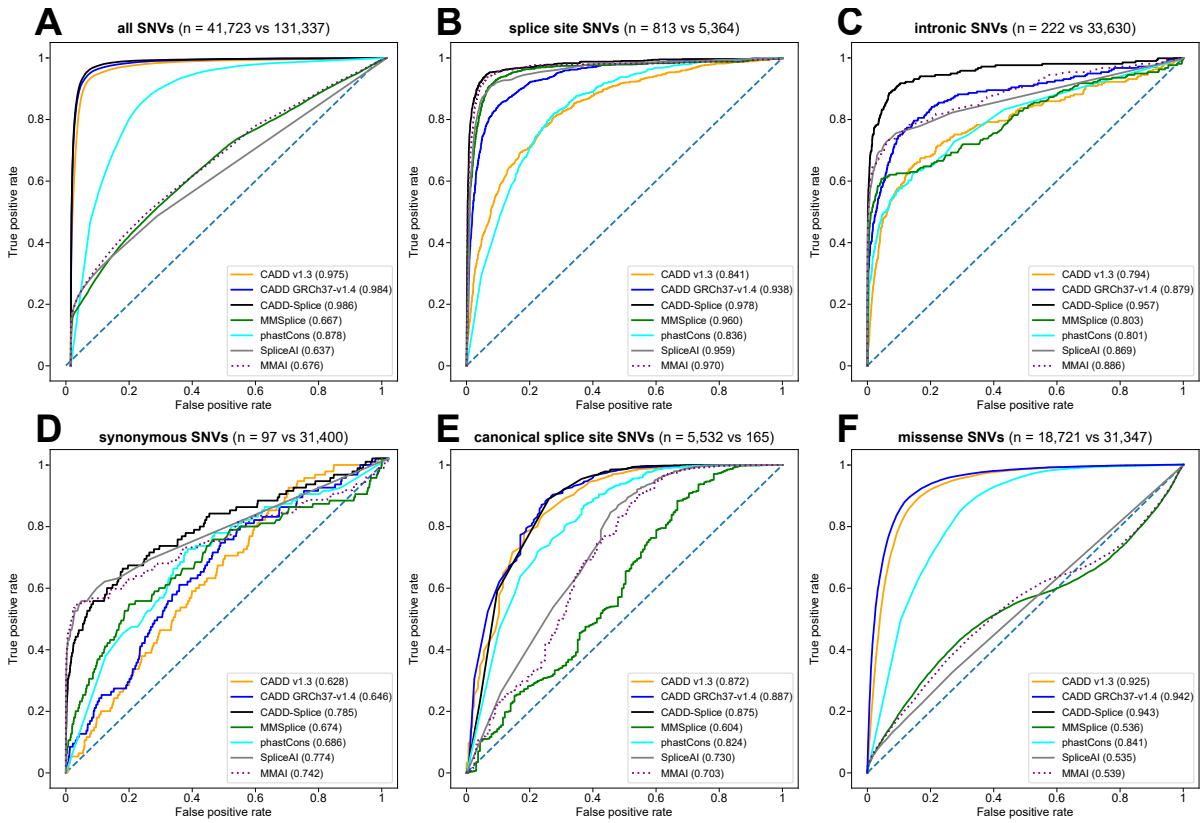

Figure 39: **Improvement on splicing related variants in comparison of ClinVar pathogenic and common gnomAD variants.** ROC benchmark of CADD models and the new splicing features on separating ClinVar pathogenic SNVs from gnomAD variants with minor allele frequency greater 0.05. Including the entire data set in the benchmark (**A**), the three CADD versions achieve the highest auROC values. CADD-Splice performs distinctly better than CADD v1.3 and CADD GRCh37-v1.4 when the benchmark is limited to splice site (variants within 20 bp of a known splice site **B**), intronic (**C**) or synonymous SNVs (**D**). CADD GRCh37-v1.4 leads on canonical splice site SNVs (**E**), while on missense SNVs (**F**), CADD-Splice and CADD GRCh37-v1.4 have an almost identical ROC.

7.4 CADD-Splice for GRCh38

In the same way that CADD GRCh37-v1.4 was extended to CADD-Splice, SpliceAI and MMSplice were added as features to the previous release for GRCh38, CADD GRCh38-v1.5. CADD GRCh38-v1.6 further corrects the GERP (Chapter 6.4) and Ensembl Regulatory Build (Zerbino *et al.*, 2015) annotations. For the first time, model parameters used for training the models are the same for GRCh37 and GRCh38. Both models are trained for 13 iterations using the same scikit-learn implementation of L-BFGS logistic

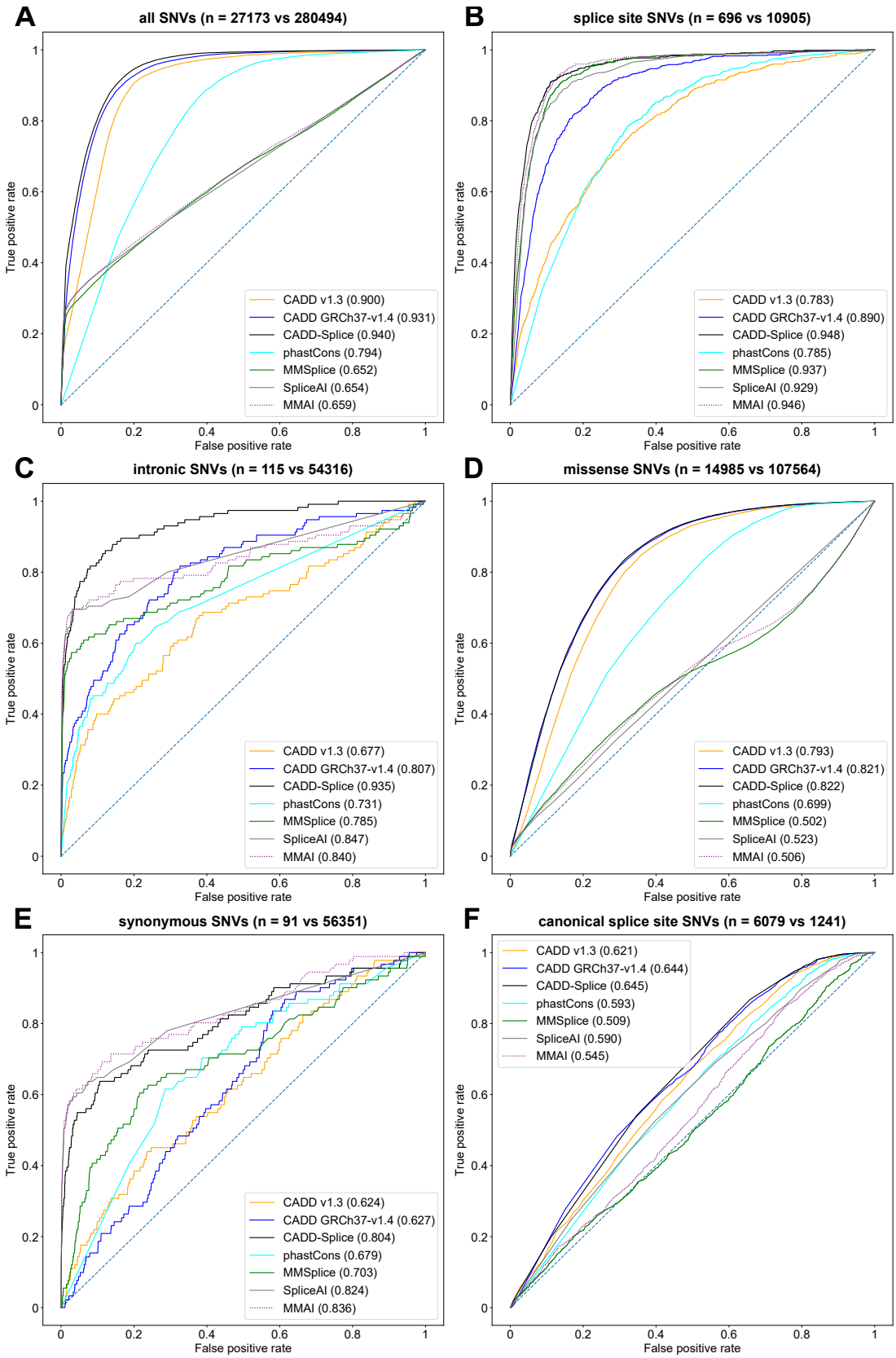

Figure 40: **ClinVar likely-pathogenic in comparison to rare gnomAD variants.** *Description on next page.*

Figure 40: *Figure on previous page.* The benchmark data set of pathogenic variants from ClinVar and common variants from ExAC or gnomAD has been frequently used for testing and training of variant classifiers. An alternative benchmark can be created by measuring ROC of likely-pathogenic variants from ClinVar in comparison to rare (MAF < 0.05, allele count > 1) variants from gnomAD. Similar to the findings in Fig. 39, CADD-Splice reaches a higher auROC for all SNVs (**A**), splice site (**B**), intronic (**C**) and synonymous SNVs (**E**), while only marginal increases in auROC are detected for missense (**D**) and canonical splice site SNVs (**F**).

regression with the regression parameter $C = 1$. For benchmarking, scored variants are lifted from GRCh38 to GRCh37 using CrossMap (Zhao *et al.*, 2014) to enable the comparison to CADD-Splice.

CADD GRCh38-v1.6 has similar performance improvements over the previous release as CADD-Splice for the ClinVar pathogenic versus common gnomAD benchmarks. Increases in auROC are observed for splice site and intronic SNVs (Fig. 42B&C) while predictions of all and only missense SNVs (Fig. 42A&D) have almost no change in auROC.

Pearson correlation between CADD GRCh38-v1.5 and CADD GRCh38-v1.6 is 0.96 for both 100,000 genome-wide SNVs and the subset of 10,079 SNVs with a Phred-score greater than 10 in CADD GRCh37-v1.4 (Fig. 37). With that, the correlation is lower than between the last two GRCh37 releases. A possible explanation are the adjustments to the GERP annotations specific to GRCh38. Those may also explain why CADD GRCh38-v1.6 is slightly more correlated to any GRCh37 release than CADD GRCh38-v1.5 (i.e. CADD GRCh37-v1.4 to CADD GRCh38-v1.5 0.79 and 0.81 versus CADD GRCh37-v1.4 to CADD GRCh38-v1.6 0.80 and 0.81). On the same 100,000 SNVs, the GerpS annotation from GRCh37 has a Pearson correlation coefficient of 0.32 with the erroneous GerpS from CADD GRCh38-v1.5, while the correlation is 0.69 with the fixed GerpS from CADD GRCh38-v1.6. However, no statistical analysis was run on this question and the observed increase may as well be due to fluctuation in model coefficients.

7.5 Conclusions from extending CADD with splicing annotations

When genomes are investigated for variants that affect phenotype, the causal molecular process is usually unknown. Therefore, variant effect scores need to integrate knowledge

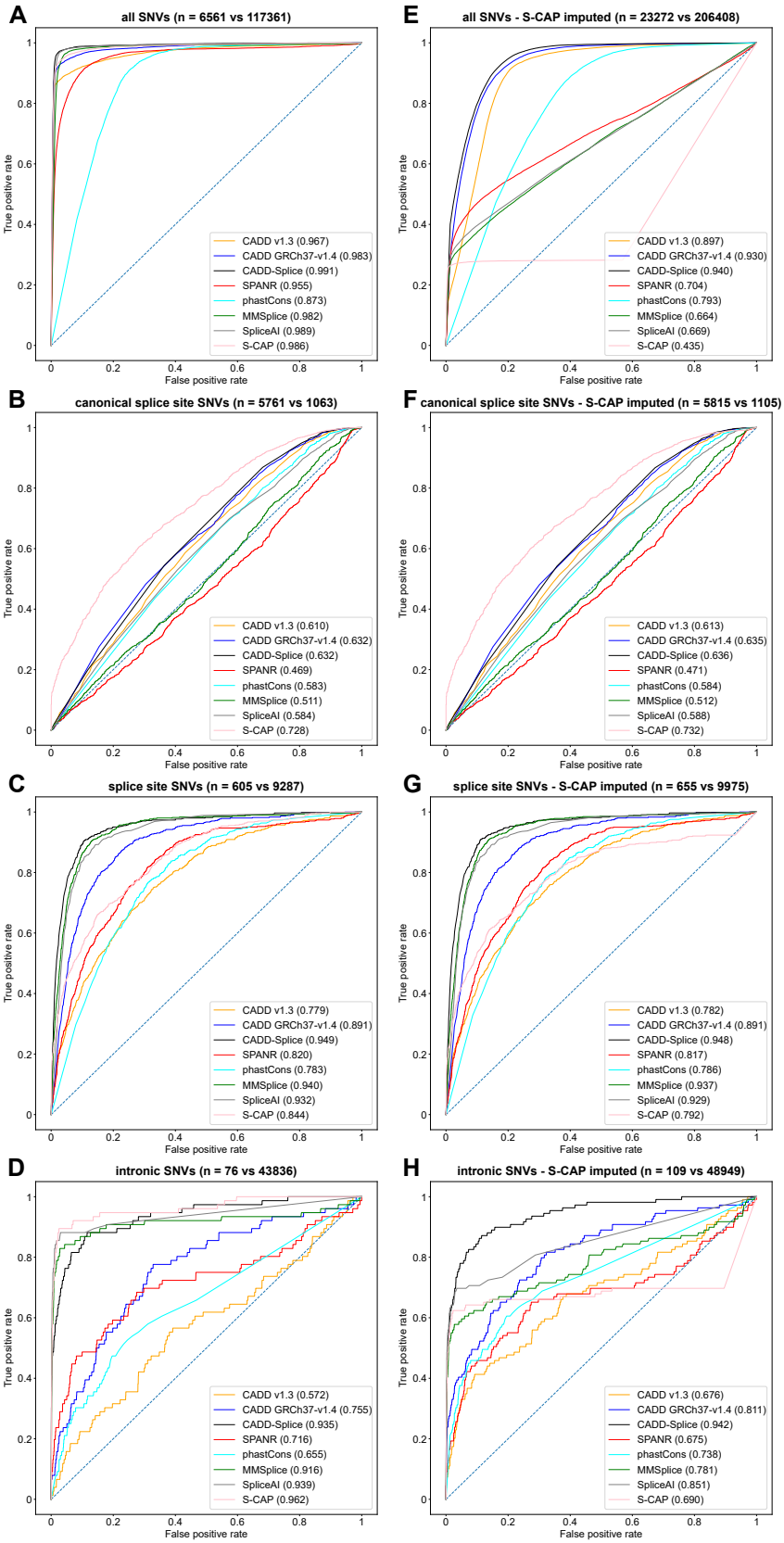

Figure 41: **Additional scores on ClinVar likely pathogenic in comparison to rare gnomAD variants.** *Description on next page.*

Figure 41: *Figure on previous page.* The specialized splicing predictors S-CAP and SPANR overlap in their training setting with the ClinVar pathogenic set and are therefore benchmarked on ClinVar likely-pathogenic variants. While both predictors have lower auROC than CADD on all (**A**) and splice site SNVs (**C**), S-CAP achieves the highest performance for SNVs located in canonical splice sites (**B**) and introns (**D**). The number of variants in the benchmark is reduced since S-CAP misses many variants. According to the S-CAP manuscript, missing variants should be classified as benign. If missing variants are imputed as such, the auROC of S-CAP decreases for all (**E**), splice site (**G**) and intronic SNVs (**H**). For canonical splice site variants where the number of variants not scored by S-CAP is small, the auROC is less affected (**F**). The auROC of other methods in is generally not affected by the change in variant numbers.

across different processes in order to rank variants across variant consequences, e.g. amino acid substitutions, truncating variants, and splicing alterations. To our knowledge, existing predictors scoring all types of genomic variants do not specifically take RNA splicing effects into account, as evident by their limited performance on specialized data sets (Mather *et al.*, 2016; Li *et al.*, 2017; Cheung *et al.*, 2019). This chapter has demonstrated that sequence-based models of splicing effect can improve the performance of existing annotation-based variant effect scores. Specifically, we have shown that the integration of deep learning derived scores from MMSplice and SpliceAI into CADD enables splice effect prediction with high accuracy.

We have benchmarked available splice predictions on the experimental MFASS data set and on known disease-causing mutations from ClinVar. Even though MFASS does not cover some types of variants like gain-of-function mutations and deep intronic variants, it is a very valuable data set for splicing prediction and the most comprehensive data set for experimental splice-site effects today. We were able to show that existing splice models work well in predicting splice effects, provided that tools use the genomic context of each variant and not the assay-specific sequence design as input for the prediction. It further benefits methods when they are not only available as a precomputed score but provided as software that can be run genome-wide and independent of reference genome build and other annotations. We note that the performance of all methods differs between exonic and intronic sequence (as expected due to different levels of constraints), as well as with distance to the canonical splice site. Even CADD v1.3, which uses only a 20 bp distance to canonical splice sites, has high precision in distinguishing pathogenic variants at

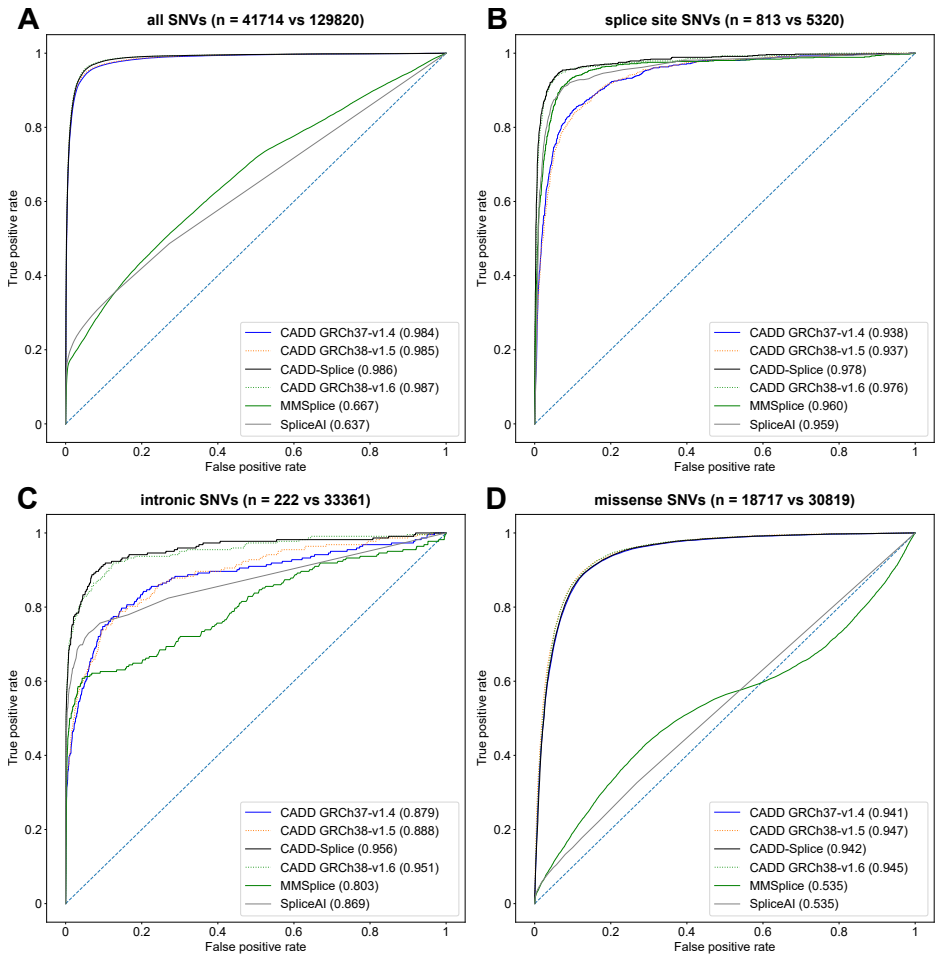

Figure 42: **CADD-Splice for GRCh38.** ROC of pathogenic SNVs from ClinVar in comparison to SNVs with MAF > 0.05 from gnomAD. CADD GRCh38-v1.6 has a higher auROC on all SNVs in the two data sets (**A**), splice site SNVs (**B**) and intronic SNVs (**C**) compared to the previous release CADD GRCh38-v1.5 while maintaining a similar auROC for missense SNVs (**D**).

canonical splice sites and shows reasonable performance for intronic variants. Based on the results of the benchmark sets, it is unknown how far we can generalize observations for intronic variants that are more than 40 bp away from a known splice junction as such variants are not included in the MFASS data set and are rarely discovered from disease studies (Mort *et al.*, 2014).

Of note, our findings contradict the original MFASS publication (Cheung *et al.*, 2019) that found HAL among the best performing predictors. We show that including psi as a feature provides an assay-specific predictive advantage and that without this feature, HAL's performance is comparable to MMSplice and SpliceAI. While part of this obser-

vation is probably due to biases of the assay, i.e. that certain exons are more frequently integrated into the reporter construct than others, some of it could be biological signal. More specifically, it could be argued that prevalent splice junctions (high psi) are less susceptible to disruption than less prevalent ones where multiple alternatives are generated. It has been previously observed that mutation effects scale non-monotonically with the inclusion level of an exon, with mutations having maximum effect at a predictable intermediate inclusion level (Baeza-Centurion *et al.*, 2019). It was suggested that competition between alternative splice sites is sufficient to cause this nonlinear relationship. We thought about integrating this in our model but could not determine a sensible feature. The pext score that was investigated as a genome-wide psi substitute, did not capture splice effect size.

The two models released as version 1.6 present the currently latest iteration of CADD. The GRCh37 model CADD-Splice has a total of 1,029 features derived from 102 annotations. 222 features X_n derive from 90 numerical annotations and one-hot-encoding of 12 categorical Boolean annotations. 14 Boolean indicators W_n express whether a given feature or feature group (out of cDNApos, CDSpos, protPos, aminoacid-substitution, targetScan, mirSVR, Grantham, PolyPhenVal, SIFTval, Dist2Mutation, chromHMM, dbscSNV-ada, dbscSNV-rf, and SpliceAI) is undefined. 12 nucleotide and 189 amino acid substitutions possible to create with SNVs correspond to another 201 features. Further, 16 different variant consequence categories and a set D consisting of the 37 annotations bStatistic, cDNApos, CDSpos, Dst2Splice, GerpN, GerpS, mamPhCons, mamPhyloP, minDistTSE, minDistTSS, priPhCons, priPhyloP, protPos, relcDNApos, relCDSpos, relProtPos, verPhCons, verPhyloP, Dist2Mutation, freq100, freq1000, freq10000, rare100, rare1000, rare10000, sngl100, sngl1000, sngl10000, the four SpliceAI scores (acceptorgain, acceptorloss, donorgain, and donorloss) and the five MMSplice scores (acceptorIntron, acceptor, donorIntron, donor and exon) are used to create a set of 592 consequence interactions. The entire model fitted using the logistic regression implementation in

scikit-learn is:

$$\begin{aligned}
&\beta_0 + \sum_{i=1}^{222} \beta_i X_i + \sum_{i=1}^{4} \sum_{j=1}^{3} \gamma_{ij} \mathbb{1}_{\text{i-th Ref category and j-th Alt category}, i \neq j} \\
&+ \sum_{i=1}^{189} \delta_i \mathbb{1}_{i\text{-th amino acid exchange possible in SNV}} + \sum_{i=1}^{14} \tau_i W_i \\
&+ \sum_{i=1}^{16} \sum_{j \in D} \alpha_{ij} \mathbb{1}_{i\text{-th Consequence category}} X_j
\end{aligned} \tag{1}$$

For CADD GRCh38-v1.6 the number of total features is 1,028 derived from 120 annotations. A full list of annotations included in CADD v1.6 is described in the release notes on our website and the supplement of our latest manuscript Rentzsch *et al.*, 2021.

8 An alternative training data set

The success of the CADD method can be attributed to two main achievements: First, the large training data set of more than 30 million variants that are not dependent on any prior classified and therefore possibly biased set of variants. And second, the large number of annotations that can be used with such a large training data set. While other projects such as CAPICE (Li *et al.*, 2020) have used features very similar to CADD with another training data set, it has been a question for some time whether there are other suitable data sets that can be used to train a variant interpretation model. One option for such a data set are variants derived from population sequencing studies that are divided into classes by minor allele frequency (MAF) in the human population.

In this project, SNVs and small InDels are split into sets based on allele frequencies. Very similar to the CADD approach, variant simulations of these sets match the nucleotide substitution patterns and genome-wide distribution. The generated variant sets then serve as training instances for a CADD-like classifier using the same annotations and model parameters as CADD GRCh38-v1.6. Finally, the new models are compared in their performance to predict pathogenic variants.

8.1 Selecting variants by allele frequency

Variant allele frequencies are not created entirely by chance. For any variant appearing in an individual, the same restrictions as for the human ancestral sequence apply. Deleterious variants that may arise in a single individual are likely to be selected against and are therefore expected at lower allele frequencies than neutral variants that may proliferate in the population. Selected for only by drift, it has been found deleterious alleles are younger than neutral alleles of the same frequency and more likely to decrease in frequency in the near future (Kiezun *et al.*, 2013).

Recent studies have made genome-wide variant calls from ten thousands of unrelated humans available. The 1000 Genomes project expected to observe at minimum 95% of SNVs that appear in more than 0.5% of individuals (The 1000 Genomes Project Consortium, 2015). With the increased number of genomes involved in more recent population studies like gnomAD (Karczewski *et al.*, 2020), NHLBI BRAVO/TopMed (Taliun *et al.*, 2021) and NCBI ALFA (Phan *et al.*, 2020), variant allele frequency estimates have extended further to less frequent variants.

This analysis is based on gnomAD version 3.0, as published on October 16th, 2019

at `https://gnomad.broadinstitute.org`. Alternative sources of population variants not used here would be gnomAD version 2.1.1 for the previous human reference build GRCh37 and the BRAVO freeze 8 release from the TOPMed consortium. gnomAD v3.0 consists of variants called from WGS data of 71,702 unrelated individuals that were mapped to GRCh38. The study includes individuals of many different subpopulations, specifically highlighting humans with African and African-American, Amish, Latino and admixed American, Ashkenazi Jewish, East Asian, Finnish, Non-Finnish European, and South Asian ancestries. In total, there are 602 million SNVs and 105 million InDels in the data set. The data was downloaded in one bgzip-compressed VCF file. For each variant, this file contains the general VCF variant information (chromosome, position, dbSNP ID, reference, and alternative sequence), as well as information about the gnomAD variant filtering, variant allele frequency (over all samples and frequency in predefined human subpopulations), allele count (over all subpopulations), number of individuals with sufficient coverage at the variants position, numbers of homozygous and heterozygous allele counts and much more.

8.2 Training set preparation

All variants were filtered based on the filtering criteria described in the VCF file. In addition, InDels longer than 50 bp were filtered out. Variants located on alternative haplotypes, chromosome Y, and the mitochondrial genome were excluded from the analysis to prevent inconsistencies in annotations. After filtering, the data set contains 525 million SNVs and 68 million InDels. A large majority of these variants are observed only in a few individuals, with 51.1% of SNVs and 41.3% of InDels being detected exactly once (Fig. 43). On the other side of the spectrum, 2.8% of SNVs and 6.5% of InDels have allele counts (AC) of more than 1,000. Due to the large total number of variants, these apparently small fractions include millions of individual variants. In order to correct for regional biases that affect the total number of haplotypes called at each genomic position, allele frequency is a better descriptor than allele count for these high-frequency variants. The variant distribution by minor allele frequency (MAF, Fig. 44) is very similar to that of allele counts, with more than 85% of SNVs and 70% of InDels being observed with MAFs of less than 1 in 10,000. About 1.8 million SNVs and 340,000 InDels are found with MAFs greater than 50%, meaning that the reference allele is the actual minor allele. Allele frequencies specified for human subpopulations are not analyzed.

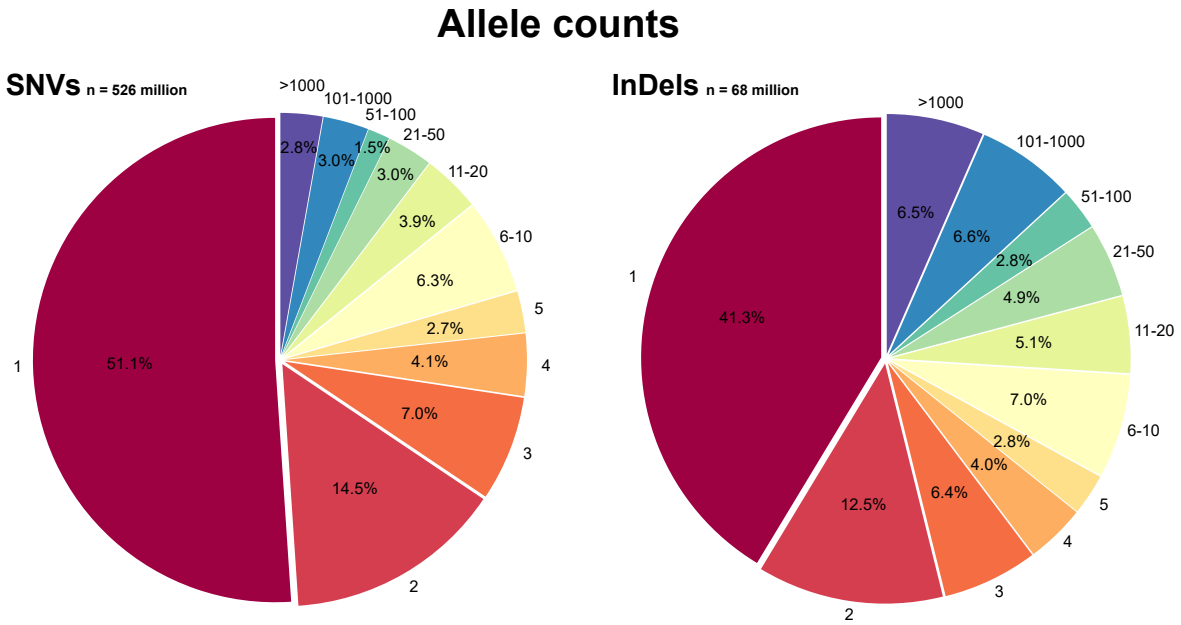

Figure 43: **Distribution of allele counts in gnomAD.** In total there are 71,702 individual genomes or 143,404 haplotypes in gnomAD v3.0. Most variants appear in a small fraction of these alleles with 3.0% of SNVs and 6.7% of InDels having an allele count of more than 1,000. More than 40% of both SNV and InDels are singletons.

Based on these frequency distributions, variants were split into four groups. The largest group are all variants that were detected once as a haploid allele in the entire data set (singletons: AC = 1). The smallest group are variants with MAF greater than 0.5 (inverse). All variants between these two groups are split into frequent and rare variants (frequent: MAF $\geq$ 0.001, rare: MAF $<$ 0.001, AC $>$ 1). While the split threshold between frequent and rare is arbitrary and not based on any biological meaning, it leads to about 26 million SNVs in the frequent group, a number approximately twice as large as the number of SNVs used in the CADD training categories.

Frequent, rare, and singleton variants were simulated based on genome-wide variant statistics via the same process that has previously been described for CADD (Kircher *et al.*, 2014): A list of 819 gaps in the GRCh38 reference genome was downloaded from `http://hgdownload.cse.ucsc.edu/goldenPath/hg38/database/gap.txt.gz`. 272 gaps that are located on chromosome Y or alternative haplotypes like chr19_GL949746v1_alt and missing from many annotations were excluded. Genome segments were defined as the sequence between subsequent gaps. Segments longer than 200 kb were split into consecutive sequence windows with a length of at minimum 100 kb.

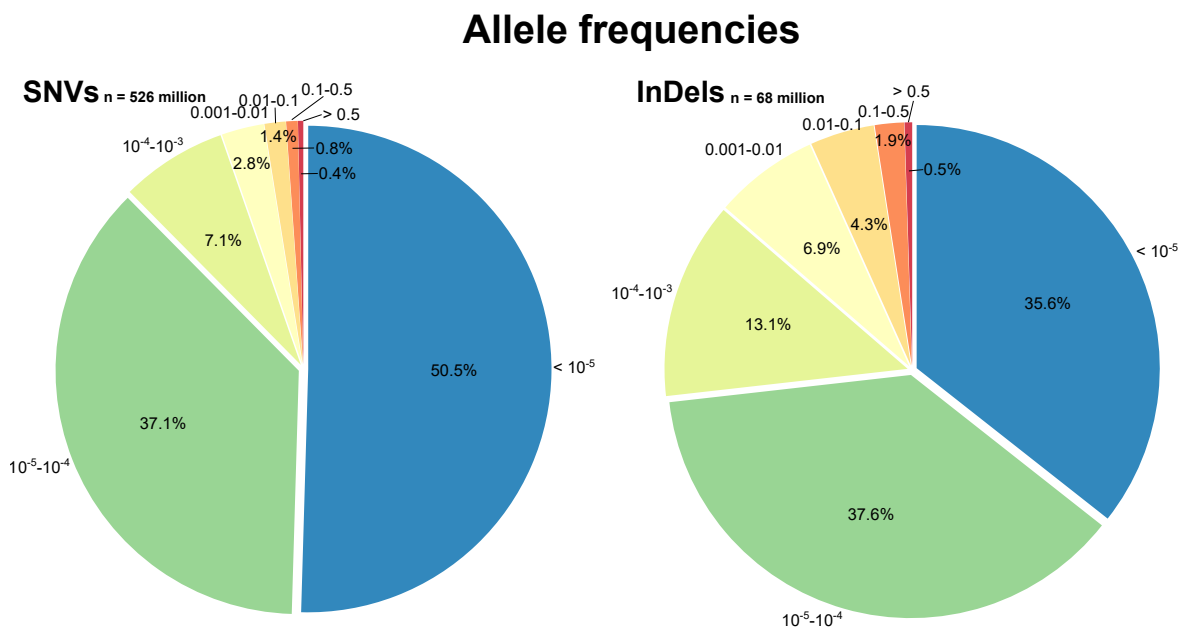

Figure 44: **Distribution of Minor Allele Frequencies in gnomAD.** Most of the 526 million SNVs and 68 million InDels are found only in a small minority of samples. For 0.4% of SNVs and 0.5% of InDel (1.8 million SNVs and 342,009 InDels) the alternative allele is actually the major allele with an allele frequency higher than 50%. 5.0% of SNVs and 13.1% of InDels have MAFs between 0.5 and 0.001.

The obtained segments were used to adjust for local differences in mutation rates. For each segment, GC content, the total number of InDels, and the number of SNVs detailed per reference nucleotide were reported. In order to control for CpG-specific mutational processes, CpGs and CpG variants were counted separately. While segmental counts are per reference nucleotide (i.e. number of Ts mutated), the SNV numbers of each substitution type (i.e. number of T to A mutation) were counted per chromosome.

Variants of the three frequency groups were simulated based on the generated statistics. Two simulations were performed to account for different recombination dynamics: one for chromosome X and one for the autosomes in which case the chromosome-wide counts were joined. We define variant probability as the likelihood of a possible substitution at any position in the reference sequence to be part of the data set. Variant probabilities were calculated for every substitution type of reference and alternative as the fraction of reference-base nucleotides where that substitution type is observed in the frequency group. An expected variant count is defined for each chromosome segment as the count of each nucleotide times the variant probabilities of each possible substitution,

summed over all nucleotides. For each segment, a local variant burden was calculated by dividing the number of observed variants of each type by the expected count.

A sampling rate parameter was used to manage the number of variants generated. Simulated SNVs were generated by iterating over all nucleotides, and selecting each possible substitution with the likelihood of that substitutions variant probability (or CpG substitution if the nucleotide is in a CpG) times local variant burden times the sampling rate. Multiple variants could be chosen for every genomic position. In order not to randomly obtain fewer variants than in the input data set even though variants were evaluated and selected individually, an excess of variants was simulated via an increase of the sampling rate. The final variants were then randomly selected from the larger set.

InDels were generally simulated like SNVs. InDel variant probabilities were calculated from counts and total sequence length, separately for insertions and deletions of each length. Expected InDel count per segment was calculated as the total nucleotide count times the sum of all InDel variant probabilities. The simulated InDels were generated by iterating over all nucleotides and selecting variants at each position with a variant probability of each insertion and deletion length, also multiplied by local InDel burden and sampling rate. While deletions are constraint by the reference sequence, inserted sequences are random strings of all four nucleotides.

The original human-derived and the six generated sets were analyzed for depletion of highly conserved variants in comparison to the CADD simulation (Fig. 45). Among all variants, the human-derived set contains the largest amount of variants with low conservation and while variants with high conservation are strongly depleted. Similar, but weaker, enrichment/depletion is observed for the variant sets frequent and rare. The singleton-sim set is, in comparison to the CADD simulation, depleted of lowly conserved variants and increased for highly conserved variants.

The substitution profile of a variant set is the relative fraction of variants with each type of substitution, i.e. combination of reference and alternative allele. Complementary combinations, like G to A and C to T, are considered jointly in such a profile. The substitution profiles of the four frequency groups are very different (Fig. 46&47). While rare and frequent SNVs substitution profiles are similar with about 40% C to T (including G to A) and 27% T to C, singletons have similar fractions of about 30% for either of these. For InDels, analyzed via lengths distribution instead of substitutions (Fig. 47), the numbers of insertions and deletions are almost the same in the frequent group

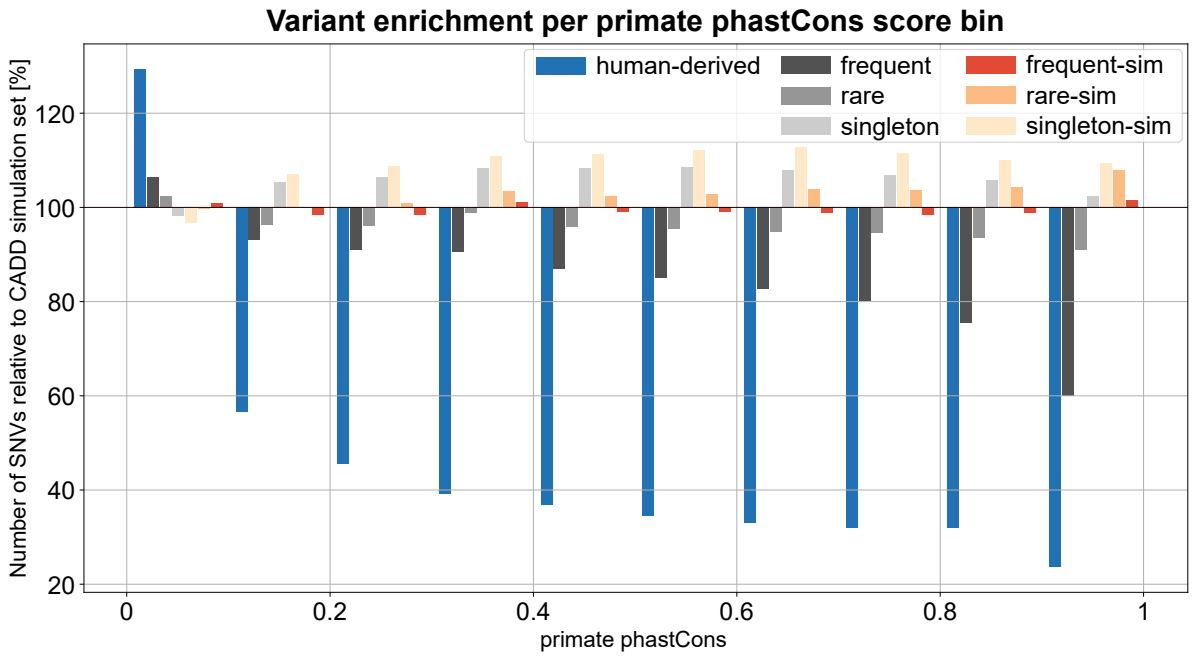

Figure 45: **Enrichment of phastCons scores in different variant sets compared to the simulated set from CADD.** 10,000,000 SNVs from each set are grouped into 10 bins based on the primate phastCons conservation score. For each bin, the number of variants in each set is compared to the set of simulated variants from the CADD training set. Bins greater than 100% are enriched while bins less than 100% are depleted relative to the simulation. For example, human-derived variants are enriched in the bin of lowest conservation and increasingly depleted with higher conservation. Bins do not indicate absolute variant numbers, with a majority of variants in each set falling in the bin of lowest conservation ranging from 0 to 0.1.

while only about 40% of rare and singleton InDels are deletions. Besides the nucleotide triplet shift that separates in-frame from frame-shift variants, reference and alternative sequences of InDels are rarely informative for assessing the pathogenicity of a variant. Nevertheless, the substitution is used as a feature in CADD to offset nucleotide biases inherited from other features. The differences in the substitution profiles may therefore be a problem when training models that use two frequency sets as positive and negative class respectively.

It would be simple to omit the substitution feature from such a model. However, to reduce the number of differences in the model structure compared to CADD, variants from the rare and singleton sets are matched to the substitution profile of the frequent set. This is done by using the larger size of the two sets and sampling each type of

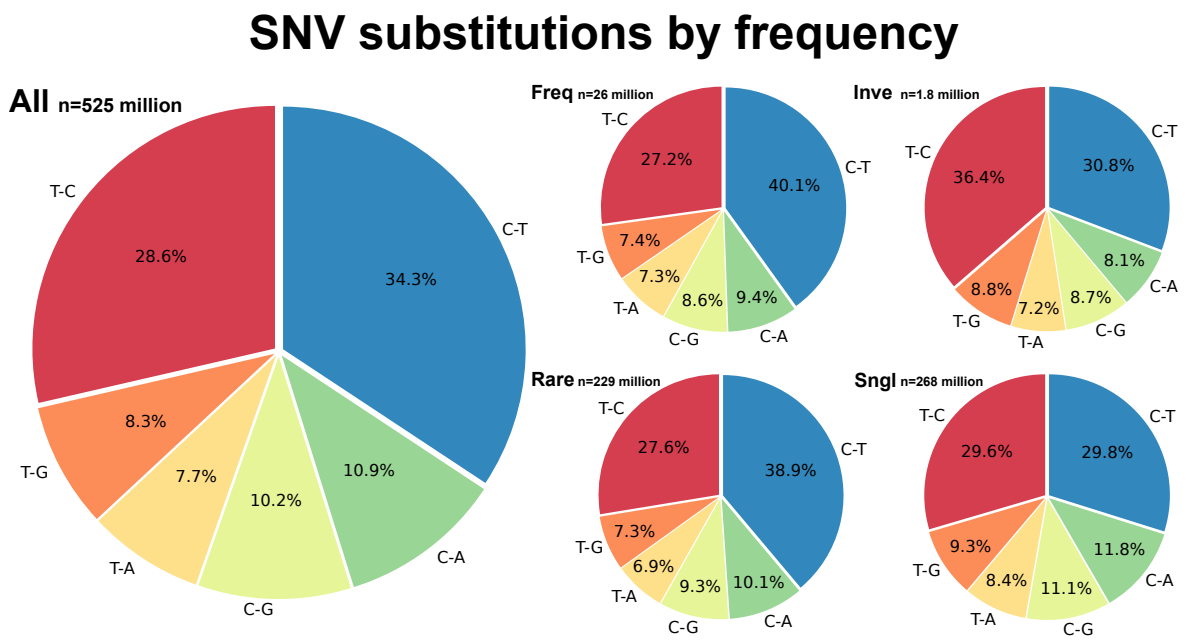

Figure 46: **SNV substitution frequencies in different allele frequency groups.** The frequency of different SNV types based on references-alternative substitution depends on the allele frequency. SNVs with cytosine/guanine (C) as reference nucleotide are more likely to be in the frequent (**Freq**, $0.5 >= MAF > 0.001$) or rare (**Rare**, $MAF <= 0.001$, $AC > 1$) variant set than the entire gnomAD variant set (**All**). Singleton (**Sngl**, $AC = 1$) variants are enriched for variants with adenine/thymine as reference base. Completely opposite of the other frequency sets, inverse (**Inve**, $MAF > 0.5$) variants have more T to C than C to T variants. Variants with adenine or guanine in the reference position are joined with their complement.

substitution separately. For SNVs, the 6 possible substitutions in non-CpG nucleotides (compare Fig. 46) and three substitutions in CpG nucleotides (C to G, C to A, and C to T, with G substitutions counted on the - strand) are sampled separately while for InDels, each length between 1 and 50 nucleotides is sampled separately for insertions and deletions. The two created variant sets are called rare-matched and singleton-matched.

In summary, nine novel variant sets were generated via splitting, simulation, and matching (Tab. 7). No data set of inverse variants was generated due to the relatively small size of the group and the ambiguous definition of variants with MAF greater than 0.5 as minor allele. In order to perform model training with equally sized classes, all data sets were randomly subsampled to the number of variants in the smallest group (frequent: 26,255,876 SNVs, 8,864,604 InDels).

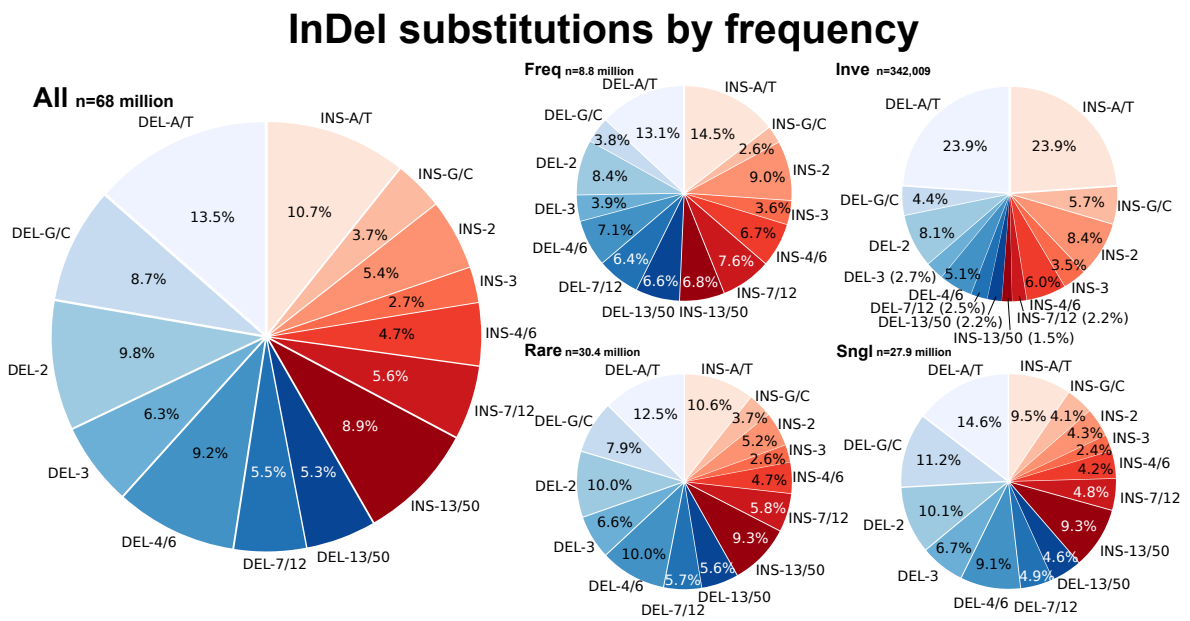

Figure 47: **InDel substitution frequencies in different allele frequency groups.** Similar to SNVs, the relative numbers of small insertions and deletions vary by allele frequency. For frequent (**Freq**) and inverse (**Inve**) variants, insertions and deletions are found in similar quantities. In the rare (**Rare**) and singleton (**Sngl**) group, there are more deletions than insertions. For single nucleotide InDels, thymine and adenine as well as cytosine and guanine in the reference (deletions) or alternative (insertions) position are joined.

8.3 Training models

Using combinations of the described data sets as input, 11 models were trained (Tab. 8). Included are the three variant sets frequent, rare, and singleton each in comparison to the respective variant simulations. Further models were trained between the three variant sets. Not all of the positive-negative set combinations were initially envisioned but are motivated later by findings as outlined below. Notably, the variant substitution matching was motivated by the finding that the combination frequent versus rare (`fr`) sharing a similar substitution profile leads to better benchmark results than the model trained from frequent versus singleton (`fs`), which do have very different substitution profiles. The final model shown here, `sim-only` was trained by contrasting simulated variants of the sets frequent and rare and serves as a control. For each data set, a 1% random hold-out was excluded for testing. Each training set contains approximately 69,538,000 variants, compared to 31,085,862 in CADD GRCh38-v1.6. Variations in data set size occur because the hold-out is determined by selecting each variant with a

Table 7: Novel training data set

frequent variants from gnomAD v3.0 with MAF > 0.001

rare variants from gnomAD v3.0 with MAF ≤ 0.001, AC > 1

singleton variants from gnomAD v3.0 with allele count of 1

frequent-sim substitution and distribution matched simulation of **frequent**

rare-sim substitution and distribution matched simulation of **rare**

singleton-sim substitution and distribution matched simulation of **singleton**

rare-matched variants from **rare**, selected by substitution frequency of **frequent**

singleton-matched variants from **singleton**, selected by substitution frequency of **frequent**

probability of 1%. All models were trained via the CADD pipeline and share the model parameters of CADD GRCh38-v1.6 of 13 training iterations with L_2-regression penalty $C = 1$ using the L-BFGS logistic regression algorithm implemented in scikit-learn.

Table 8: Models trained from different positive and negative training sets

Identifier	positive set	negative set
`f`	frequent	frequent-sim
`r`	rare	rare-sim
`s`	singleton	single-sim
`fr`	frequent	rare
`fs`	frequent	singleton
`rs`	rare	singleton
`frmatched`	frequent	rare-matched
`fsmatched`	frequent	singleton-matched
`rsmatched`	rare-matched	singleton-matched
`frmutden`*	frequent	rare
`sim-only`	frequent-sim	rare-sim
CADD*,†	human-derived	simulation

* includes mutation density annotations
† CADD GRCh38-v1.6

Model training accuracies were determined by predicting all variants in the respective training set with the developed model. The baseline against which all new models are compared is CADD GRCh38-v1.6, which has an accuracy of 62.2%. Based on the lower level of depletion in any of the population variant data sets, it was expected that the training accuracy of any population-variant model would be lower than that.

We observed that an initial batch of models, which used the same annotations and features as CADD GRCh38-v1.6, reached accuracies higher than observed for CADD. `frmutden`, an example for such a model, has a model accuracy of 62.9% (Fig. 48A). An analysis of the model coefficients found, in comparison to all other model coefficients, increased weights of the mutation density annotations. Combined, the mutation density annotations were responsible for 75% of the score variance observed on the 1% hold-out from the training set. The largest coefficients were learned for the features of variant count in a 100 bp window around the variant for frequent and rare variants. While these variant counts per genome window are not variant-specific, are not based on gnomAD but release 5 of the BRAVO variant server, and mask the presence or absence of specific variant positions, it has to be assumed that there is some circularity between the annotations and the training set. Without the mutation density annotations, accuracies are between 52% and 58%. `f`, the model derived from frequent variants has the highest accuracy of the three models of observed variants versus simulation while `s`, the model derived from singleton variants has the lowest accuracy of that group. Similarly, the training accuracy of `fs`, the model trained from frequent versus singleton variants, is higher than that of `fr` and `rs`, the models trained from rare variants in comparison to either frequent or singleton. Both findings indicate an association between class frequency and the previously observed enrichment levels in the three classes frequent, rare, and singleton. The three models trained with substitution matched variant sets, `frmatched`, `fsmatched` and `rsmatched`, have lower training accuracies than their unmatched counterparts. `rsmatched` has the lowest accuracy of all trained models of 52.3%, followed by `sim-only` (52.4%). Due to the high number of instances in the data sets, differences between training accuracies greater than 0.1% are significant.

Model correlations are calculated from the scores of 100,000 SNVs that had been randomly selected genome-wide (Fig. 48B). Spearman correlations are calculated from the raw model predictions of (set I) all SNVs and (set II) all 8,849 SNVs with a Phred-score greater than 10 in CADD GRCh38-v1.6. Very high correlations of more than 0.95 of set I are found between `f` and `r`, `r` and `s`, in all combinations between `fr`, `frmatched`

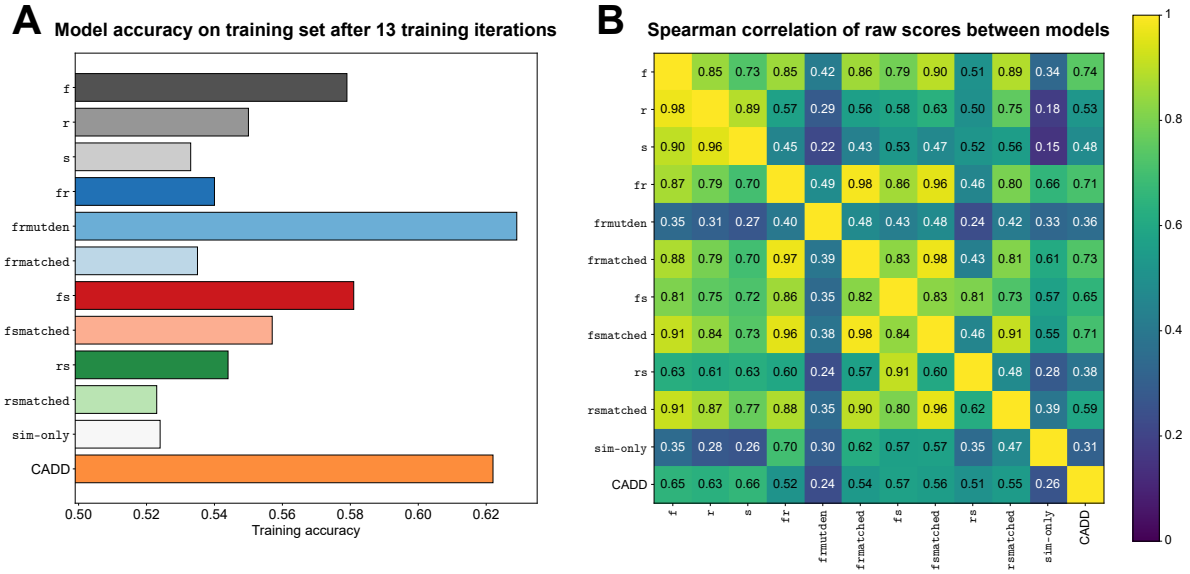

Figure 48: **Training accuracy and model correlation with different training set combinations.** 11 models, as specified in Tab. 8, were trained based on different combinations of the developed training data set. (**A**) Accuracy on correctly classifying instances from the training set was measured after training for 13 L-BFGS iterations. Model accuracies are between 52% and 63%, all within 0.1% of the respective accuracy on a 1% random hold-out from the training set. Only the model *frmutden*, which uses mutation density annotations that overlap with the training set, has a higher training accuracy than CADD GRCh38 v1.6. (**B**) To characterize the differences between models, Spearman correlation coefficients of raw model scores are calculated based on predictions of 100,000 randomly selected SNVs (bottom-left), as well as those 8,849 SNVs from the first set with a Phred-score greater than 10 in CADD GRCh38-v1.6 (top-right). While all models are positively correlated, the models *frmutden* and *sim* do have correlations of less than 0.5 with the majority of all other models.

and `fsmatched`, and between `fsmatched` and `rsmatched`. On set II, correlations greater than 0.95 are only observed in all combinations between `fr`, `frmatched` and `fsmatched`. The other highly-correlated combinations from set I are nevertheless correlated between 0.85 and 0.91. On both sets, `fsmatched` and `rsmatched` are more correlated to the other matched models than to their unmatched counterparts. While no negative correlations are observed, `frmutden` has correlations of less than 0.5 with all other models on both sets I and II. `sim-only` has correlations of less than 0.5 with seven other models on both sets I and II. The lowest overall correlations are also observed for `sim-only` with correlation coefficients of 0.26 on set I and 0.15 on set II compared to `s` and 0.28 on set

I and 0.18 on set II compared to `r`. The latter is especially noteworthy since `sim-only` and `r` use the same negative training set rare-sim. Aside from `frmutden` and `sim-only`, only `s` and `rs` are correlated less than 0.5 with any model on any of the two sets.

CADD GRCh38-v1.6 is moderately correlated with most models aside `frmutden` and `sim-only`, with correlations coefficients between 0.51 and 0.66 on set I and 0.38 and 0.74 on set II. On set I, correlations of more than 0.6 are observed with the three frequency group versus simulation models `f`, `r` and `s`, while on set II, correlations higher than 0.7 are observed with `f`, `fr`, `frmatched` and `fsmatched`. All correlations are lower than compared to the latest GRCh37 model of CADD (Fig. 37).

The eleven population-variant models were scored on the same benchmarks as previous CADD releases. Since model development is still in the model testing and optimization phase, all ClinVar benchmarks were not run on all chromosomes but limited to the even-numbered chromosomes and chromosome X. The same hold-out had previously been established CADD releases since version 1.4. On pathogenic variants from ClinVar in comparison to common (MAF > 0.05) variants from gnomAD (Fig. 49), the best performing new models (`f`, `fr`, `frmatched` and `fsmatched`) have slightly better auROC on all variants than CADD. When limited to SNVs, CADD is the best model in this benchmark with an auROC of 0.987 compared to 0.986 for `f`, `fr` and `frmatched` and 0.984 for `fsmatched`. In comparison to that, CADD is not the best model for InDels with an auROC of 0.985 compared to 0.989 to 0.991 for the four mentioned new models. All of the highest performing new models use the frequent data set as its positive class. The frequent class (MAF > 0.001 in whole-genome sequencing of gnomAD 3.0) arguably overlaps with the positive class in the above-mentioned benchmark (MAF $>$ 0.05 in exome sequencing of gnomAD 2.1, lifted to GRCh38), which may lead to inflated results for the models that are using it as a training set. To estimate this effect, all models were benchmarked to separate the same pathogenic variants from ClinVar from benign variants of the same database (Fig. 50), as well as ClinVar variants classified as likely-pathogenic from those classified as likely-benign (Fig. 51). The findings on these benchmarks in terms of auROC are highly similar to the first comparison in that CADD is performing best for SNVs, while a number of the new models outperform CADD on InDels. It was further noted that variant matching, though increasing auROC for SNVs for both `fsmatched` compared to `fs` and `rsmatched` compared to `rs`, did not increase but rather decrease auROC for InDels in all three benchmarks. Further, in all three subsets, `sim-only` did have a higher auROC than random while `frmutden` model has a

worse auROC than `fr` on SNVs but the same auROC on InDels.

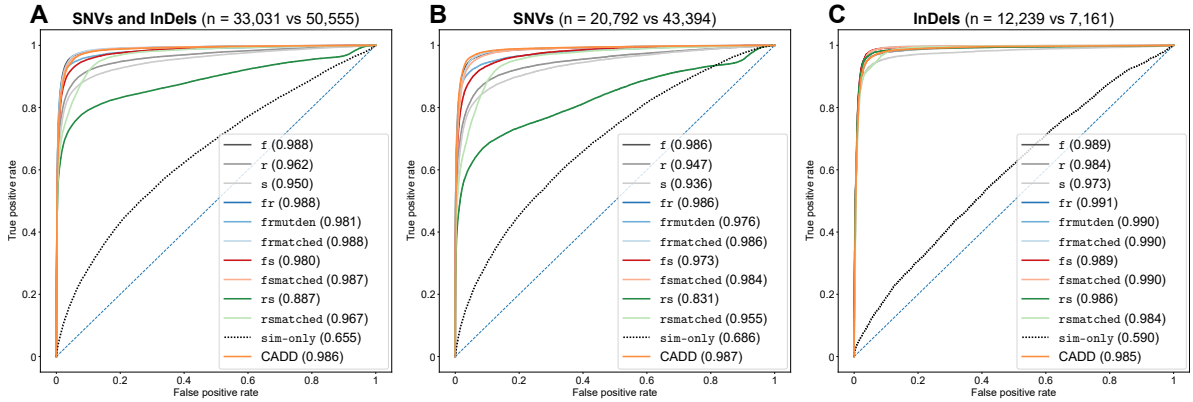

Figure 49: **ClinVar pathogenic in comparison to common gnomAD variants.** ROCs of variants classified as pathogenic in ClinVar compared to common (MAF > 0.05) variants in gnomAD for SNVs and InDels (**A**), as well as separately for only SNVs (**B**), and only InDels (**C**). Only variants on even numbered chromosomes and chromosome X are scored.

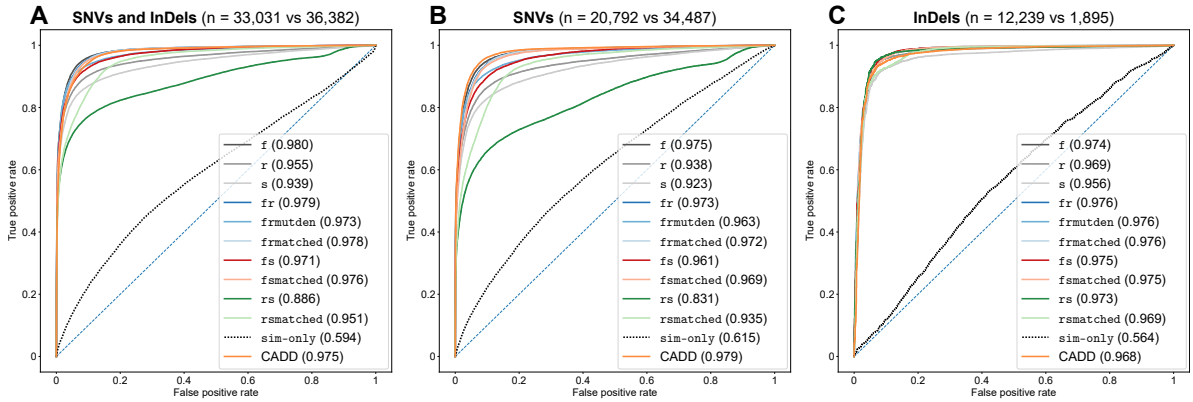

Figure 50: **ClinVar pathogenic in comparison to ClinVar benign variants.** ROCs of variants classified as pathogenic in ClinVar compared to those classified as benign in the same database. Panels show SNVs and InDels (**A**), only SNVs (**B**), and only InDels (**C**). Only variants on even numbered chromosomes and chromosome X are scored.

To assess model performance on protein-coding variants, SNVs tested in DMS assays were used as a benchmark by correlating model prediction and assay readout (Fig. 52). The four data sets adapted here are assays of E3 ligase activity of BRCA1 (Starita *et al.*, 2015), PDZ3-domain ligand binding of DLG4 (McLaughlin *et al.*, 2012), thiopurine methylation activity of TPMT (Matreyek *et al.*, 2018) and WW-domain substrate

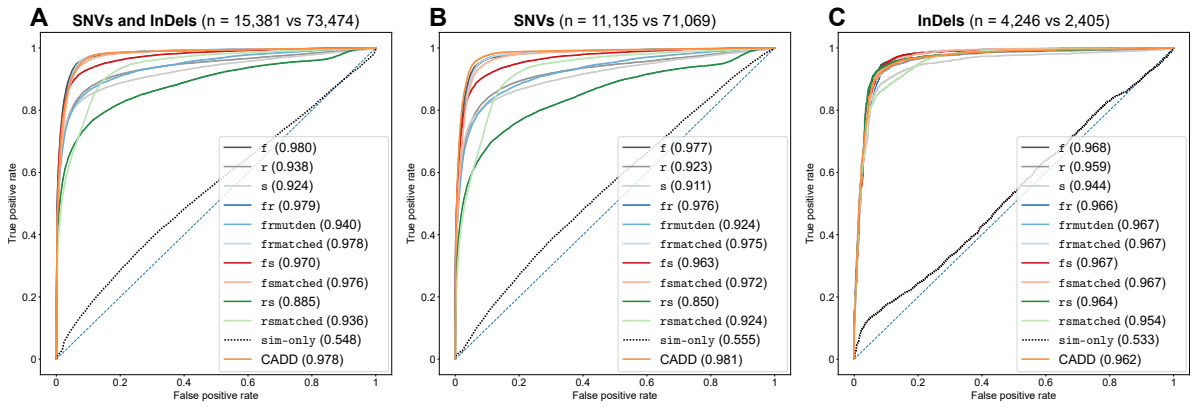

Figure 51: **ClinVar likely-pathogenic in comparison to ClinVar likely-benign variants.** ROCs of variants from the ClinVar database, contrasting those classified as likely pathogenic to those classified as likely-benign. Panels display SNVs and InDels (**A**), only SNVs (**B**), and only InDels (**C**). Only variants on even numbered chromosomes and chromosome X are scored.

binding of YAP1 (Fowler *et al.*, 2010). All data sets were filtered for protein amino-acid substitutions that can be caused by one or more SNVs, with one SNV randomly chosen in cases where multiple SNVs would lead to the same amino acid substitution. In all cases, negative correlations are better since high model scores predict the functional impact and low assay scores arise when the function of a protein is disrupted. Correlations are considered separately for each protein. CADD has the highest (negative) Spearman correlation coefficients for the benchmark set of DLG4 and TPMT, while `frmatched` leads on those of BRCA1 and YAP1. No significant improvement is observed in the substitution matched models as `fr` and `frmatched` perform very similar and for `fs` and `rs`, the matched models perform better for BRCA1 but worse for DLG4 and YAP1. On TPMT, `fs` correlates slightly better than `fsmatched`, while `rsmatched` correlates better than `rs`. As for the ClinVar benchmark, the model `f` has the highest correlation of the three frequency group versus simulation models, while `r` has a higher correlation than `s` in three of the four proteins, with the exception being the smallest data set of 180 YAP1 variants. Of all tested models, `sim-only` has the lowest correlation with all four DMS assays. `frmutden` has considerably lower correlations than `fr` in all four benchmarks.

8.4 Conclusions from training alternative models

As already concluded in previous chapters, the variation in conditions and selection criteria of multiple benchmarks can favor different CADD models. It is therefore not

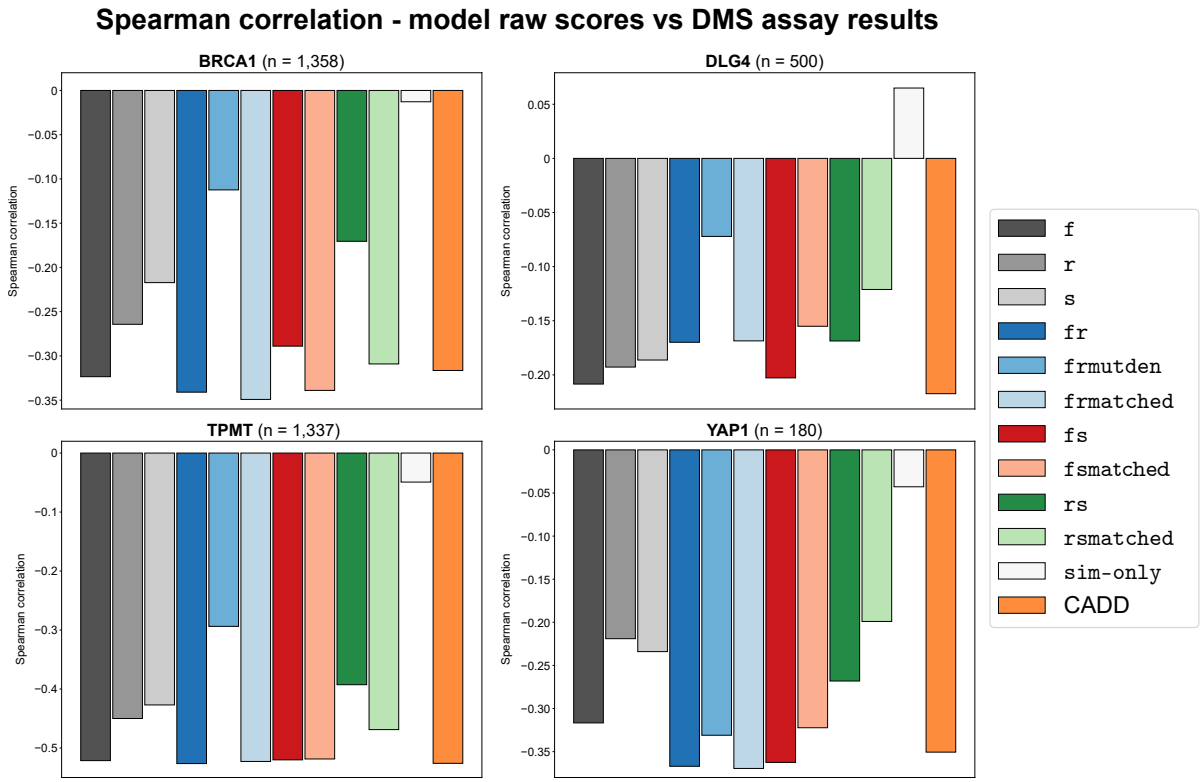

Figure 52: **Predicting deep mutational scanning results.** Spearman correlation of predictions of the eleven population-variant models and CADD GRCh38-v1.6 compared to assay readouts from DMSs of four human protein-coding genes. Negative correlations are better since high model predictions (i.e. assumed pathogenicity) should relate to low DMS values for assays like domain enzymatic activity or substrate binding.

surprising that it is impossible to select the one model trained from the alternative training data sets that performs best on all four DMS data sets. Due to the small size of the benchmark sets (the YAP1 data set consists of 180 SNVs) and the fact that all variants are highly clustered in a small genomic window, the model coefficients of a single annotation may have large effects on overall model performance. It is currently unknown if this is causing inflated results for some models and should be clarified by larger and more diverse data sets. Nevertheless, we conclude that frequency binning is a suitable method to define sets of variants that are increasingly depleted of pathogenic or deleterious variants. This is seen over all benchmarks as the model `f` generally performs better than `r` which again generally performs better than `s`. The size of the training set used here was defined by the number of frequent variants in gnomAD. There are more than 200 million SNVs of each, the rare and singleton group, in gnomAD which would

enable an even larger training set from those two groups.

While the depletion of potentially deleterious effects is lower in the three frequency groups and hence the trained models have even lower training accuracies than CADD, any of the trained models has an auROC higher than expected by pure chance on distinguishing between known pathogenic and potentially neutral variants. That is remarkable especially for `sim-only` which was trained from contrasting simulated variants of frequent and rare variants. While different levels of enrichment were observed, it is apparently enough to learn a model that separates between pathogenic and neutral variants. The auROC by `sim-only` is far smaller than that of other models and less than single annotations like conservation scores. `sim-only` also has the lowest correlations to the DMS data set. It remains possible that the random variant selection based on genome-wide distribution leads to an enrichment in only a few annotations, as was found for primate phastCons, but is not enriched or depleted in most other annotations. We speculate that the different phases of chromatin could be enriched in the sequence windows around frequent compared to rare variants. Nevertheless, the results clearly show that the variants in the CADD simulation set are not entirely randomly selected from the genome. Entirely random selection of variants from the entire genome remains not advisable since not all regions of the genome do have the same level of sequence quality and may lead to unfounded biases in the training set. A possible solution may be the selection of variants genome-wide in regions associated with high-confidence mapping by the Genome in a Bottle Consortium (Zook *et al.*, 2019). The idea of using different frequency groups for training mitigates the same problem insofar as all variants have been identified through the same pipeline, although at different allele frequencies. The obtained benchmark results are very positive, with multiple models exceeding CADD in auROC on the ClinVar benchmark. This is in contrast to the results for SNVs where CADD has the largest auROC.

It was found that the matching in the substitution frequencies in the models that train on variants of multiple frequency groups generally decreases training accuracy and increases model correlation, in between population-variant models and in comparison to CADD. For `fr` and `frmatched`, the differences between models and model results are generally smaller than is the case for `fs` or `rs`, most likely due to the smaller difference in substitution frequencies in the unmatched model `fr`. Model performance especially increases for `fsmatched` and `rsmatched`, though some DMS benchmarks are higher for the unmatched models. Whether this is due to the small number of instances in the test

sets or due to better representation of amino acid exchanges in the unmatched sets may be clarified by more comprehensive test sets. Notably, the auROC of all three ClinVar benchmarks on the subset of InDel variants decreases for `rsmatched` compared to `rs`. As the auROC of the same benchmarks does barely increase for the other matched models and due to the finding that even frequent InDels are often not found across multiple large sequencing projects, InDel matching may need to be revised further.

With a maximum overall correlation of 0.66 (Fig. 48B), the observed correlations to CADD GRCh38-v1.6 of any new model is much lower than between versions of CADD (Spearman correlation of 0.96 between GRCh38-v1.5 and GRCh38-v1.6, Fig. 37), and even lower than across genome references (Spearman correlation of 0.80 between GRCh37-v1.6 and GRCh38-v1.6). So far, it could not be determined which annotations are responsible for this difference. The highest overall correlation with the three frequency group versus simulation models (`f`, `r` and `s`) suggests that some of this is due to the selection of variants from the simulation. The slightly higher correlations (greater 0.7 with `f`, `fr`, `frmatched` and `fsmatched`) on variants with a Phred-score greater than 10 indicates that the differences are founded in annotations that are less relevant for scoring deleteriousness. In any case, the comparable performance of novel less correlated models on the benchmarks may be suggesting that the overall good results are not due to learning the same model from another training set.

8.5 Possible improvements in annotations compared to CADD

The trained models are only a first step in the process of creating models similar to CADD based on standing variation. In that, it has been prerogative to keep the model as similar as possible to CADD. However, data sets used for training the population-variant models add new constraints regarding which annotations can be used as model features. That some adaptions are necessary has been shown in the case of the `frmutden` model, which has a much higher training accuracy than all other models including CADD but is less correlated to other models and generally results in lower benchmark scores than the `fr` model that is different in only the mutation density annotations. There may be more biases in the annotations that have not yet been found, especially regarding the InDel variants as we are still missing large InDel benchmarks.

Similar constraints arise from the CADD training data and limit which and how annotations are used to train the CADD model. For example, the conservation scores phastCons and phyloP are specifically generated for CADD by excluding the human

reference sequence from the score calculation. This had been necessary since the human-derived variants would all be recent changes in the human reference compared to the other aligned sequences and therefore have decreased conservation score values. With the human-derived variants out of the training set, it becomes possible again to use "default" conservation scores that include the human sequence. Preliminary findings have induced caution about such a step as phastCons scores generated by including the human sequence have different boundary limits than those without human reference (the end of the sequence alignment in phastCons with human is converging to 0.25 while phastCons without human converges to 0, while both have same lower conservation score limit of 0) and may need extended feature processing and evaluation. In how far including the human genome, which is the result of a recent population bottleneck event, negatively affects any conservation score may also have to be evaluated carefully.

Another big advantage of using standing variation over human-derived variants is that the former are all defined as variants from the reference genome, while the latter are variants where the genome evolved into the reference sequence. In VCF file format, the reference sequence is always that of the reference genome. In the annotation of the CADD training set, human-derived variants are therefore annotated in reverse, i.e. with reference and alternative switched. The switch causes that any stop-gain variant becomes a stop-loss variant and reverse. As the CADD simulation set is generated so that it matches the human-derived set, but arriving from the reference sequence, the switch has to be compensated by switching the annotated variants back in the encoding step. However, some annotations are lost in the switch. The most prominent example are start-loss variants since the opposite of a start-loss variant would be a start-gain and it is not possible to annotate start-gain variants with the currently available methods. In CADD, this is overcome by masking start-loss variants in the training set, as well as in model application, as missense variants. Other annotations where switching is applied are scores such as MMSplice and SpliceAI, where a prediction from the reference sequence is subtracted from a prediction from the alternative sequence. The subtraction in those scores is reversed in the human-derived set. This may cause an annotation bias when the score distribution over all variants is not symmetrical, i.e. the difference between scores of reference and alternative sequence is unequal from zero. Asymmetry has previously been observed for MMSplice (Fig. 35) and was one of the reasons that lead to the exclusion of DNA shape factors in CADD v1.4 (Tab. 5). With the new training data set, such switching is not necessary, enabling the annotation of start-loss variants and

simplifying the processing of other annotations.

Any new model will not necessarily be a replacement for CADD. Instead, it may only be a slightly different weighting of the same annotations, which reduces outlier predictions based on artifacts in one or a few features. The independent feature weights may also put additional focus on different types of variants like non-coding effects, transcription start sites, or insertions and deletions. Alternatively, the new model could use a completely separate but not necessarily uncorrelated list of annotations that generates independent results and may increase or decrease the level of suspicion generated by CADD for variants of unknown significance. Whether entirely complement or only adding several features that can not be used in CADD, such a model may finally be integrated with CADD in a single, even larger, meta-classifier.

8.6 Similarity to other projects

Using variants from population sequencing projects has been used in the development of many variant scoring projects. Many classifiers use variant allele frequency as assessment of pathogenicity. Others, including Eigen, FitCons, LINSIGHT, CDTS, CCR (Havrilla *et al.*, 2019) and CAPICE, use standing variation as part of their training set to establish a variant classifier. Novel in this case is the split of variants into frequency groups and then using these groups as class labels for training a supervised classification.

Similar to how CADD has been adapted for mouse, pig, and chicken, the population-variant approach could be adapted for other organisms. With the current technology, population sequencing projects like gnomAD are very expensive and it is hard to imagine that thousands of individual WGS data sets may be generated for any non-human animal or plant. However, another drop in sequencing cost and the economic incentives involved in breeding projects such as highly prized cattle may lead to smaller but similar projects for economically important domestic animal and crop species in the future.

Discussion

The objective of this thesis was to advance genome-wide variant effect scoring. In the first three chapters, I have outlined the fundamentals of machine learning and genomics, followed by an introduction to variant effect scoring. Based on these foundations and the observation that existing methods are complicated to improve on, I laid out considerations to develop variant effect scores using the principles of reproducibility, extensibility, and simplicity. These considerations lead to the implementation of a workflow that enables fast and reliable development of variant effect scores. The workflow is applied to the previously established variant effect score CADD. After evaluation of different machine learning algorithms, performance benchmarks, and genomic annotations, it leads to the development of the first variant effect score based entirely on the human reference genome build GRCh38 and greatly improved model predictions of RNA splicing effects. Finally, I describe a novel variant effect score training data set of variants derived from population sequencing whose properties could enable improved variant classification in the future.

A new workflow increases flexibility regarding model development

When this project began in 2017, there were already a large number of machine learning based methods for predicting all kinds of variant effects. In discussions with many different people, we found that there was a lot of potential to address shortcomings in all of these methods. Based on the fundamentals and existing solutions described in chapter 3, I have developed a generalized concept for variant effect scoring. This concept especially focuses on a number of critical components that I found sparsely described in many other publications. In particular, I describe precomputed and *ad hoc* annotations, scoring of multiple predicted outcomes for a single variant, and how SNVs and InDels can be annotated in a joined evaluation. In addition, chapter 4 covers technical aspects like sparse matrix notations that enable larger training matrices and discusses the necessary steps between the trained machine learning model and the applied variant effect score.

The developed workflow described in chapter 5 implements all these principles. Originally designed to replicate the previous iterations of CADD, four Snakemake pipelines manage training set annotation, model training and benchmarking, genome-wide SNV scoring, and finally user-defined variant scoring. All processing is split into multiple, consecutively executed steps and interacts with a job scheduler. File access times and

processing resources depend on the specific file system and working memory limits. Accordingly, our implementation is optimized for the computational infrastructure at the Berlin Institute of Health. It relies only on open source software, which provides the highest guarantee that the entire process runs on any compute infrastructure or can be adopted with minor adjustments. The individual components of the underlying framework are similar to existing programs. Unique is the workflow that integrates pre-computed and *ad hoc* annotations in a single model, versioned feature selection based on configuration files, and parallel training and benchmarking of multiple models. Full flexibility in terms of annotations, features, machine learning algorithm, training, and benchmark data sets allowed us to explore many different settings. Automating the entire process from training set to scored genomic variant greatly simplified the generation of score releases. For example, a collaborator once requested a CADD score release that was excluding a single annotation. We were able to realize such a project, that would previously have been dismissed for being too complex to be worth the effort, with minimal hands-on time in approximately two weeks. In conclusion, the new capabilities of the introduced variant effect scoring workflow significantly simplified and accelerated the generation of the subsequently presented results.

CADD performs well on GRCh38

We leveraged the capabilities of the new workflow to reconsider several decisions that had been made in the original development of CADD, including training data set composition, machine learning algorithm, and which annotations contribute to model performance. However, the main quest of improving CADD was developing the first variant effect score for the latest human reference genome GRCh38, as is described in chapter 6. We note that variant effect scores specific to the previous reference GRCh37 had been established for many years and there was no indication that the same would not be possible on other references. Instead, the main obstacle that had previously prevented the development of such a score was obtaining all the required data, i.e. annotations and training set. For the initial GRCh38 release of CADD (GRCh38-v1.4), we generated these data either by repeating the required analyses or obtaining matching replacements. We find that all developed CADD versions are strongly correlated, both within and across genome references. On all performed testing and validation benchmarks, the GRCh38 models are similar in performance to the respective GRCh37 models. Such a high degree of similarity across genome references is expected since comparisons are

limited to genome regions where variants (or scores) can be converted between references. Benchmarks are also typically selected from well-studied genomic regions where GRCh38 models are unlikely to benefit from better annotation quality. We note that current CADD models are not able to take advantage of all improvements in GRCh38. For example, variants on alternative haplotypes can currently not be scored as not all annotations are defined in those regions. Nevertheless, the annotations and models benefit from numerous fixes and generally higher quality of the GRCh38 reference sequence. As the final step in a variant analysis pipeline, the lower error rates obtained during read-alignment and variant calling to GRCh38 lead to more confidence in the evaluated variants and ultimately fewer false positive results.

Specialized annotations increase performance and improve the prediction of splice effects

In chapter 7, I presented the results of extending CADD with annotations that are predictive of RNA splicing effects. As a result of my analysis of an experimentally derived splice variant set, the deep learning based scores MMSplice and SpliceAI are integrated as annotations into CADD. Benchmarking results indicate that models which specifically predict splicing effects are superior for identifying splice-altering variants when the only possible variant effect is an alteration of RNA splicing. However, the causal molecular process is unknown in most research or clinical applications of variant effect prediction. In these cases, where only a difference in phenotype has been observed and different molecular consequences have to be evaluated, annotation-based variant effect scores like the developed CADD-Splice model perform better than the specialized scores. Remarkably, the improved performance of the combined scores is observed even when the comparisons are limited to splice proximal or intronic variants. We speculate that this is due to a combination of the annotated splicing effects and features of species conservation. The latter contradicts the conclusion by Cheung *et al.*, 2019 that conservation scores are almost uncorrelated with splicing effects, a finding that we could attribute to a processing error in the study. Similarly, we could not support the previous finding that measures of splice-isoform abundance are predictive of variant effects and would attribute the reported correlation to assay-specific effects. Our findings suggest that variant effect scoring may generally be improved by integrating process-specific information. We believe that this finding is universal and may improve prediction in many non-coding genomic regions where current variant effect scores are of limited precision.

Population-derived variants provide a larger and potentially more universal training data set

The novel training set presented in chapter 8 further expands the scope of data sets used to train variant effect scores. As demonstrated by benchmarking results similar in performance to CADD, we find that low training accuracies of between 52% and 58% are not necessarily indicative of ineffective variant effect scoring. However, we find that a single annotation that increases training accuracy can significantly decrease model performance, serving as an important reminder that novel annotations have to be tested rigorously before they are added to a model. This observation is similar to that from chapter 6, where nonlinear models achieved higher training accuracy but lower benchmark performance. Both results highlight the importance of independent testing and validation. We conclude that while a large training data set may suffice for training a good variant effect model, all involved components have to be evaluated carefully. With nearly 70 million variants in the training data set, each of the novel models shown uses more than twice as many training instances as CADD, demonstrating the scalability of our approach. As population sequencing efforts are generating larger data sets every year, this number may be scaled up further. The most interesting aspect of the novel training sets is that they do not consist of variants that entered the current genome by mutation from an ancestral sequence, while being equally distributed genome-wide. The novel data set may hence enable us to adopt annotations for which the current CADD training data set is biased.

Outlook

CADD is a widely used method for genomic variant scoring. This is reflected in more than 100 weekly variant scoring requests on the CADD web servers, numerous support requests via email, and the fact that CADD has been a pipeline component, source of information, inspiration as well as a method to improve upon, in many other scientific publications. As of November 2021, the first two CADD manuscripts (Kircher *et al.*, 2014 and Rentzsch *et al.*, 2019) have been highlighted as high impact publications in Web of Science and have accumulated, according to Google Scholar, more than 5,000 citations. The third manuscript (Rentzsch *et al.*, 2021) complements these works and describes the process that led to the development of CADD-Splice.

This is not meant to imply that CADD and other variant effect scores are perfect

and their development is finished. Besides bug fixes and ideas for improvements to the machine learning, training data set, or additional web services provided, there is an almost endless list of possible variant features. Given that general pathogenicity scores such as CADD include annotations from many different sources, developers have to carefully assess the quality of each of those and ensure that features are not inherently biased due to how they were generated. We are hopeful that community standards such as the upcoming Matched Annotation from NCBI and EMBL-EBI (MANE) project, along with an increase in sequence-based models that can be more easily adapted to new, variant-specific annotations, will support the development of more stable, reproducible, and better predictors. The new annotations introduced in CADD-Splice (Rentzsch *et al.*, 2021) are only a first step in integrating variant-specific predictions of molecular processes like microRNA binding, polyadenylation, protein degradation, and intrachromosomal interactions. We note that prediction of gain of function events, even with highly variant-specific predictors like the deep learning based scores SpliceAI and MMSplice, is currently unreliable and will need further investigation. In this regard, our results highlight the importance of precise annotations of existing genomic elements such as splice junctions.

Being able to score SNVs and InDels genome-wide, variant effect scores like CADD can evaluate a lot more than "just" coding variants. A number of methods rely on regional CADD predictions to score large SVs (Ganel *et al.*, 2016; Kleinert & Kircher, 2021), indicating a possible path for scoring variants larger than 50 bp. Nevertheless, there are a lot of variants in regions of the genome that we are currently unable to evaluate. Heritability estimates of traits and complex diseases have found that detected variants only explain some, in extreme cases less than 10%, of the observed variance (Manolio *et al.*, 2009). While the fraction of missing heritability attributed to variants has become smaller with the inclusion of rare variants and SVs in the corresponding analyses (Génin, 2020), it raises the question of where and how the remaining phenotype affecting variants may be detected. Strong effect signals have recently been discovered to be caused by variable number tandem repeats, repetitive DNA elements which vary in copy number, that are located in protein-coding sequences (Mukamel *et al.*, 2021). However, variants in repetitive regions and mobile elements are not well-investigated as variant calling with current short-read sequencing technologies is severely limited in those genome segments. The telomere-to-telomere consortium was recently able to complete many previously unresolved genome segments using long-read sequencing and, for the first time, generated

the complete sequence of a single human genome (Nurk *et al.*, 2021). Although it is too early to assess the benefit of the additional sequences, preliminary results are promising better read mapping accuracies that could further improve annotation quality and reduce the number of false positive variant calls (Aganezov *et al.*, 2021). We note that it is currently unclear whether all of these efforts will lead to a generally accepted new genome reference. Without that, it will not only be challenging to score called variants, but require a lot of effort to obtain comprehensive and unbiased annotations.

Most variant benchmarks test the performance of different variant effect scores in isolation and classify variants based on the predicted effect alone. Clinical guidelines recommend that multiple orthogonal methods may be used to evaluate variants, with decisions being made based on majority vote (MacArthur *et al.*, 2014; Richards *et al.*, 2015). They do not, however, encourage the use of multiple variant effect scores unless those are based on independent principles. Instead, joint analysis of DNA and RNA samples has proven very effective in identifying and prioritizing splice or regulatory variants underlying differentially expressed genes (Li *et al.*, 2017; Anderson *et al.*, 2019; Mohammadi *et al.*, 2019). Tissue and cell-type specificity of most of these events requires the availability of corresponding transcriptome data. We speculate that if the option of sample-dependent data generation was feasible, computational predictions could motivate the collection of relevant tissues or the establishment of new cell lines. RNA transcript data from the patient could then be used to validate, for example, an actual splicing effect. In any case, the combination of variant effect scoring with RNA data could be very effective and should be explored more.

It is clear that the significance of individual genes for specific diseases is not well-represented in organismal and genome-wide models of variant effects (Havrilla *et al.*, 2019; Abramovs *et al.*, 2020). Existing gene-disease links may therefore aid variant prioritization in cases where such information is available. For example, information about the specific phenotype (including pathways, gene interactions, or affected tissues) is potentially of high relevance. They require careful phenotype recording via a unified classification like the Human Phenotype Ontology (Robinson *et al.*, 2008) or automated image analysis (Hsieh *et al.*, 2019; Gurovich *et al.*, 2019). Features derived from affected cell types, tissues, and the disease itself (e.g. interaction partners of known disease proteins or tissue-specific gene expression) can aid variant scoring by highlighting associated genomic regions or limiting analyses to a given gene set. Undiagnosed cases can be analyzed jointly based on similar phenotypes (Kaplanis *et al.*, 2020; Matalonga *et al.*, 2021)

and knowledge about findings in similar cases may support the interpretation of variant scores. However, the same data could also impair the discovery of less well-studied disease genes due to observation biases (Stoeger *et al.*, 2018). Given that more than 75% of human genes have never been associated with a genetic disease (Posey *et al.*, 2019), we suspect that these approaches may guide us primarily to previously detected loci, obscuring the discovery of novel disease mechanisms. Unfortunately, current model validation procedures promote this behavior by benchmarking models on well-studied disease variants. While it is obviously impossible to test performance on an unknown metric, novel annotations should preferentially be available for all instances of an effect class and be void of large systematic biases. Thus, even though other information may be useful for particular variant rankings, we remain skeptical regarding the naive integration of broad-scale annotations that prioritize variants based on their location in specific genomic regions.

Today, variant effect scores are evidently effective at predicting large variant effects, as indicated by their use in numerous scientific studies. Integrated into complex variant analysis pipelines, they are an important tool for identifying the cause of genetic diseases. Outside of human biology and basic research, variant effect scores have found application in agriculture and selective breeding. We are confident that variant effect scores can be improved further and will continue to help us understand the effects of all types of genomic variants.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., *et al.* (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org`.

Abel, H. J., Larson, D. E., Regier, A. A., Chiang, C., *et al.* (2020). Mapping and characterization of structural variation in 17,795 human genomes. Nature *583*(7814), 83–89.

Abramovs, N., Brass, A., & Tassabehji, M. (2020). GeVIR is a continuous gene-level metric that uses variant distribution patterns to prioritize disease candidate genes. Nature Genetics *52*(1), 35–39.

Abramowicz, A. & Gos, M. (2018). Splicing mutations in human genetic disorders: examples, detection, and confirmation. Journal of Applied Genetics *59*(3), 253–268.

Acuna-Hidalgo, R., Veltman, J. A., & Hoischen, A. (2016). New insights into the generation and role of de novo mutations in health and disease. Genome Biology *17*(1), 241.

Adzhubei, I. A., Schmidt, S., Peshkin, L., Ramensky, V. E., *et al.* (2010). A method and server for predicting damaging missense mutations. Nature Methods *7*(4), 248–249.

Aganezov, S., Yan, S. M., Soto, D. C., Kirsche, M., *et al.* (2021). A complete reference genome improves analysis of human genetic variation. bioRxiv *2021.07.12.452063*.

Agarwal, V., Bell, G. W., Nam, J.-W., & Bartel, D. P. (2015). Predicting effective microRNA target sites in mammalian mRNAs. eLife *4:e05005*.

Aken, B. L., Ayling, S., Barrell, D., Clarke, L., *et al.* (2016). The Ensembl gene annotation system. Database *2016*, baw093.

Alexandari, A., Kundaje, A., & Shrikumar, A. (2020). Maximum Likelihood with Bias-Corrected Calibration is Hard-To-Beat at Label Shift Adaptation. arXiv *1901.06852*.

Alexandrov, L. B., Kim, J., Haradhvala, N. J., Huang, M. N., *et al.* (2020). The repertoire of mutational signatures in human cancer. Nature *578*(7793), 94–101.

Alexandrov, L. B., Nik-Zainal, S., Wedge, D. C., Aparicio, S. A. J. R., *et al.* (2013). Signatures of mutational processes in human cancer. Nature *500*(7463), 415–421.

Alipanahi, B., Delong, A., Weirauch, M. T., & Frey, B. J. (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nature Biotechnology *33*(8), 831–838.

Amarasinghe, S. L., Su, S., Dong, X., Zappia, L., *et al.* (2020). Opportunities and challenges in long-read sequencing data analysis. Genome Biology *21*(1), 30.

Amberger, J. S., Bocchini, C. A., Scott, A. F., & Hamosh, A. (2019). OMIM.org: leveraging knowledge across phenotype–gene relationships. Nucleic Acids Research *47*(D1), D1038–D1043.

Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. (2017). Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv *1711.06104*.

Anderson, D., Baynam, G., Blackwell, J. M., & Lassmann, T. (2019). Personalised analytics for rare disease diagnostics. Nature Communications *10*(1), 5274.

Arora, S., Ge, R., Liang, Y., Ma, T., & Zhang, Y. (2017). Generalization and Equilibrium in Generative Adversarial Nets (GANs). arXiv *1703.00573*.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., *et al.* (2000). Gene Ontology: tool for the unification of biology. Nature Genetics *25*(1), 25–29.

Avsec, Z., Kreuzhuber, R., Israeli, J., Xu, N., *et al.* (2019). The Kipoi repository accelerates community exchange and reuse of predictive models for genomics. Nature Biotechnology *37*(6), 592–600.

Avsec, Z., Weilert, M., Shrikumar, A., Krueger, S., *et al.* (2021). Base-resolution models of transcription-factor binding reveal soft motif syntax. Nature Genetics *53*(3), 354–366.

Baeza-Centurion, P., Miñana, B., Schmiedel, J. M., Valcárcel, J., & Lehner, B. (2019). Combinatorial Genetics Reveals a Scaling Law for the Effects of Mutations on Splicing. Cell *176*(3), 549–563.e23.

Baird, P. A., Anderson, T. W., Newcombe, H. B., & Lowry, R. B. (1988). Genetic disorders in children and young adults: a population study. American Journal of Human Genetics *42*(5), 677–693.

Barrett, R. & Schluter, D. (2008). Adaptation from standing genetic variation. Trends in Ecology & Evolution *23*(1), 38–44.

Bermejo-Das-Neves, C., Nguyen, H.-N., Poch, O., & Thompson, J. D. (2014). A comprehensive study of small non-frameshift insertions/deletions in proteins and prediction of their phenotypic effects by a machine learning method (KD4i). BMC Bioinformatics *15*(1), 111.

Betel, D., Koppal, A., Agius, P., Sander, C., & Leslie, C. (2010). Comprehensive modeling of microRNA targets predicts functional non-conserved and non-canonical sites. Genome Biology *11*(8), R90.

Beyter, D., Ingimundardottir, H., Oddsson, A., Eggertsson, H. P., *et al.* (2021). Long-read sequencing of 3,622 Icelanders provides insight into the role of structural variants in human diseases and other traits. Nature Genetics *53*, 779–786.

Biggs, H., Parthasarathy, P., Gavryushkina, A., & Gardner, P. P. (2020). ncVarDB: a manually curated database for pathogenic non-coding variants and benign controls. Database *2020*, baaa105.

Bogard, N., Linder, J., Rosenberg, A. B., & Seelig, G. (2019). A Deep Neural Network for Predicting and Engineering Alternative Polyadenylation. Cell *178*(1), 91–106.e23.

Bouaoun, L., Sonkin, D., Ardin, M., Hollstein, M., *et al.* (2016). TP53 Variations in Human Cancers: New Lessons from the IARC TP53 Database and Genomics Data. Human Mutation *37*(9), 865–876.

Box, G. E. P. (1976). Science and Statistics. Journal of the American Statistical Association *71*(356), 791–799.

Boycott, K. M., Rath, A., Chong, J. X., Hartley, T., *et al.* (2017). International Cooperation to Enable the Diagnosis of All Rare Genetic Diseases. American Journal of Human Genetics *100*(5), 695–705.

Brown, G. R., Simon, M., Wentling, C., Spencer, D. M., *et al.* (2020). A review of inherited cancer susceptibility syndromes. Journal of the American Academy of Physician Assistants *33*(12), 10–16.

Brunak, S., Engelbrecht, J., & Knudsen, S. (1991). Prediction of human mRNA donor and acceptor sites from the DNA sequence. Journal of Molecular Biology *220*(1), 49–65.

Buniello, A., MacArthur, J. A., Cerezo, M., Harris, L. W., *et al.* (2019). The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019. Nucleic Acids Research *47*(D1), D1005–D1012.

Campbell, P. J., Getz, G., Korbel, J. O., Stuart, J. M., *et al.* (2020). Pan-cancer analysis of whole genomes. Nature *578*(7793), 82–93.

Campisi, J. & Vijg, J. (2009). Does damage to DNA and other macromolecules play a role in aging? If so, how?. The Journals of Gerontology. Series A, Biological Sciences and Medical Sciences *64*(2), 175–178.

Capriotti, E. & Fariselli, P. (2017). PhD-SNPg: a webserver and lightweight tool for scoring single nucleotide variants. Nucleic Acids Research *45*(W1), W247–W252.

Capriotti, E., Montanucci, L., Profiti, G., Rossi, I., *et al.* (2019). Fido-SNP: the first webserver for scoring the impact of single nucleotide variants in the dog genome. Nucleic Acids Research *47*(W1), W136–W141.

Cargill, M., Altshuler, D., Ireland, J., Sklar, P., *et al.* (1999). Characterization of single-nucleotide polymorphisms in coding regions of human genes. Nature Genetics *22*(3), 231–238.

Carter, H., Douville, C., Stenson, P. D., Cooper, D. N., & Karchin, R. (2013). Identifying Mendelian disease genes with the Variant Effect Scoring Tool. BMC Genomics *14*, S3.

Casper, J., Zweig, A. S., Villarreal, C., Tyner, C., *et al.* (2018). The UCSC Genome Browser database: 2018 update. Nucleic Acids Research *46*(D1), D762–D769.

Cassa, C. A., Weghorn, D., Balick, D. J., Jordan, D. M., *et al.* (2017). Estimating the selective effects of heterozygous protein-truncating variants from human exome data. Nature Genetics *49*(5), 806–810.

Castellana, S., Fusilli, C., Mazzoccoli, G., Biagini, T., *et al.* (2017). High-confidence assessment of functional impact of human mitochondrial non-synonymous genome variations by APOGEE. PLoS computational biology *13*(6), e1005628.

Chen, J., Li, X., Zhong, H., Meng, Y., & Du, H. (2019). Systematic comparison of germline variant calling pipelines cross multiple next-generation sequencers. Scientific Reports *9*(1), 9345.

Cheng, J., Nguyen, T. Y. D., Cygan, K. J., Celik, M. H., *et al.* (2019). MMSplice: modular modeling improves the predictions of genetic variant effects on splicing. Genome Biology *20*(1), 48.

Cheung, R., Insigne, K. D., Yao, D., Burghard, C. P., *et al.* (2019). A Multiplexed Assay for Exon Recognition Reveals that an Unappreciated Fraction of Rare Genetic Variants Cause Large-Effect Splicing Disruptions. Molecular Cell *73*(1), 183–194.e8.

Choi, Y., Sims, G. E., Murphy, S., Miller, J. R., & Chan, A. P. (2012). Predicting the Functional Effect of Amino Acid Substitutions and Indels. PLoS ONE *7*(10), e46688.

Chollet, F. *et al.* (2015). Keras. `https://keras.io`.

Chun, S. & Fay, J. C. (2009). Identification of deleterious mutations within three human genomes. Genome Research *19*(9), 1553–1561.

Church, D. M., Schneider, V. A., Steinberg, K. M., Schatz, M. C., *et al.* (2015). Extending reference assembly models. Genome Biology *16*, 13.

Chèneby, J., Gheorghe, M., Artufel, M., Mathelier, A., & Ballester, B. (2018). ReMap 2018: an updated atlas of regulatory regions from an integrative analysis of DNA-binding ChIP-seq experiments. Nucleic Acids Research *46*(D1), D267–D275.

Collins, R. L., Brand, H., Karczewski, K. J., Zhao, X., *et al.* (2020). A structural variation reference for medical and population genetics. Nature *581*(7809), 444–451.

Conrad, D. F., Pinto, D., Redon, R., Feuk, L., *et al.* (2010). Origins and functional impact of copy number variation in the human genome. Nature *464*(7289), 704–712.

Cooper, D. N., Chen, J.-M., Ball, E. V., Howells, K., *et al.* (2010). Genes, mutations, and human inherited disease at the dawn of the age of personalized genomics. Human Mutation *31*(6), 631–655.

Cooper, D. N., Krawczak, M., Polychronakos, C., Tyler-Smith, C., & Kehrer-Sawatzki, H. (2013). Where genotype is not predictive of phenotype: towards an understanding of the molecular basis of reduced penetrance in human inherited disease. Human Genetics *132*(10), 1077–1130.

Cummings, B. B., Karczewski, K. J., Kosmicki, J. A., Seaby, E. G., *et al.* (2020). Transcript expression-aware annotation improves rare variant interpretation. Nature *581*(7809), 452–458.

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., *et al.* (2011). The variant call format and VCFtools. Bioinformatics *27*(15), 2156–2158.

Dannemann, M. & Kelso, J. (2017). The Contribution of Neanderthals to Phenotypic Variation in Modern Humans. American Journal of Human Genetics *101*(4), 578–589.

Davis, C. A., Hitz, B. C., Sloan, C. A., Chan, E. T., *et al.* (2018). The Encyclopedia of DNA elements (ENCODE): data portal update. Nucleic Acids Research *46*(D1), D794–D801.

Davydov, E. V., Goode, D. L., Sirota, M., Cooper, G. M., *et al.* (2010). Identifying a High Fraction of the Human Genome to be under Selective Constraint Using GERP++. PLoS Computational Biology *6*(12), e1001025.

Dayhoff, M. O., Schwartz, R., & Orcutt, B. C. (1979). A model of evolutionary change in proteins. in *Atlas of protein sequence and structure* volume 5. National Biomedical Research Foundation: Washington, D.C. 345–358.

Deciphering Developmental Disorders Study (2017). Prevalence and architecture of de novo mutations in developmental disorders. Nature *542*(7642), 433–438.

di Iulio, J., Bartha, I., Wong, E. H. M., Yu, H.-C., *et al.* (2018). The human noncoding genome defined by genetic diversity. Nature Genetics *50*(3), 333–337.

Dong, C., Wei, P., Jian, X., Gibbs, R., *et al.* (2015). Comparison and integration of deleteriousness prediction methods for nonsynonymous SNVs in whole exome sequencing studies. Human Molecular Genetics *24*(8), 2125–2137.

Douville, C., Masica, D. L., Stenson, P. D., Cooper, D. N., *et al.* (2016). Assessing the Pathogenicity of Insertion and Deletion Variants with the Variant Effect Scoring Tool (VEST-Indel). Human Mutation *37*(1), 28–35.

Drubay, D., Gautheret, D., & Michiels, S. (2018). A benchmark study of scoring methods for non-coding mutations. Bioinformatics *34*(10), 1635–1641.

Dudbridge, F. (2013). Power and predictive accuracy of polygenic risk scores. PLoS genetics *9*(3), e1003348.

Duncan, L., Shen, H., Gelaye, B., Meijsen, J., *et al.* (2019). Analysis of polygenic risk score usage and performance in diverse human populations. Nature Communications *10*(1), 3328.

Echols, H. & Goodman, M. F. (1991). Fidelity Mechanisms in DNA Replication. Annual Review of Biochemistry *60*(1), 477–511.

Edwards, A. M., Isserlin, R., Bader, G. D., Frye, S. V., *et al.* (2011). Too many roads not taken. Nature *470*(7333), 163–165.

Edwards, S. L., Beesley, J., French, J. D., & Dunning, A. M. (2013). Beyond GWASs: illuminating the dark road from association to function. American Journal of Human Genetics *93*(5), 779–797.

Elhaik, E., Tatarinova, T., Chebotarev, D., Piras, I. S., *et al.* (2014). Geographic population structure analysis of worldwide human populations infers their biogeographical origins. Nature Communications *5*(1), 3513.

Elkon, R. (2003). Genome-Wide In Silico Identification of Transcriptional Regulators Controlling the Cell Cycle in Human Cells. Genome Research *13*(5), 773–780.

ENCODE Project Consortium (2012). An integrated encyclopedia of DNA elements in the human genome. Nature *489*(7414), 57–74.

ENCODE Project Consortium, Moore, J. E., Purcaro, M. J., Pratt, H. E., *et al.* (2020). Expanded encyclopaedias of DNA elements in the human and mouse genomes. Nature *583*(7818), 699–710.

Ernst, J. & Kellis, M. (2012). ChromHMM: automating chromatin-state discovery and characterization. Nature Methods *9*(3), 215–216.

Esposito, D., Weile, J., Shendure, J., Starita, L. M., *et al.* (2019). MaveDB: an open-source platform to distribute and interpret data from multiplexed assays of variant effect. Genome Biology *20*(1), 223.

Ewing, B. & Green, P. (1998). Base-calling of automated sequencer traces using phred. II. Error probabilities. Genome Research *8*(3), 186–194.

Exome Aggregation Consortium, Lek, M., Karczewski, K. J., Minikel, E. V., *et al.* (2016). Analysis of protein-coding genetic variation in 60,706 humans. Nature *536*(7616), 285–291.

Farwell, K. D., Shahmirzadi, L., El-Khechen, D., Powis, Z., *et al.* (2015). Enhanced utility of family-centered diagnostic exome sequencing with inheritance model–based analysis: results from 500 unselected families with undiagnosed genetic conditions. Genetics in Medicine *17*(7), 578–586.

Finn, R. D., Mistry, J., Tate, J., Coggill, P., *et al.* (2010). The Pfam protein families database. Nucleic Acids Research *38*, D211–D222.

Finucane, H. K., Bulik-Sullivan, B., Gusev, A., Trynka, G., *et al.* (2015). Partitioning heritability by functional annotation using genome-wide association summary statistics. Nature Genetics *47*(11), 1228–1235.

Fletcher, R. (1987). Practical methods of optimization. Wiley: Chichester, NY.

Folkman, L., Yang, Y., Li, Z., Stantic, B., *et al.* (2015). DDIG-in: detecting disease-causing genetic variations due to frameshifting indels and nonsense mutations employing sequence and structural properties at nucleotide and protein levels. Bioinformatics *31*(10), 1599–1606.

Fowler, D. M., Araya, C. L., Fleishman, S. J., Kellogg, E. H., *et al.* (2010). High-resolution mapping of protein sequence-function relationships. Nature Methods *7*(9), 741–746.

Franc, V. & Sonnenburg, S. (2009). Optimized Cutting Plane Algorithm for Large-Scale Risk Minimization. Journal of Machine Learning Research (10), 2157–2192.

Freund, Y. & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. in *Computational Learning Theory* (Vitányi, P., ed.) volume 904. Springer: Heidelberg, Germany. 23–37.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annals of Statistics *29*(5), 1189–1232.

Fu, Q., Mittnik, A., Johnson, P., Bos, K., *et al.* (2013). A Revised Timescale for Human Evolution Based on Ancient Mitochondrial Genomes. Current Biology *23*(7), 553–559.

Fu, Y., Liu, Z., Lou, S., Bedford, J., *et al.* (2014). FunSeq2: a framework for prioritizing noncoding regulatory variants in cancer. Genome Biology *15*(10), 480.

Ganel, L., Abel, H. J., FinMetSeq Consortium, & Hall, I. M. (2016). SVScore: an impact prediction tool for structural variation. Bioinformatics *33*(7), 1083–1085.

Garber, M., Guttman, M., Clamp, M., Zody, M. C., *et al.* (2009). Identifying novel constrained elements by exploiting biased substitution patterns. Bioinformatics *25*(12), i54–i62.

Geoffroy, V., Herenger, Y., Kress, A., Stoetzel, C., *et al.* (2018). AnnotSV: an integrated tool for structural variations annotation. Bioinformatics *34*(20), 3572–3574.

Giardine, B., van Baal, S., Kaimakis, P., Riemer, C., *et al.* (2007). HbVar database of human hemoglobin variants and thalassemia mutations: 2007 update. Human Mutation *28*(2), 206.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. The MIT Press: Cambridge, MA.

Grantham, R. (1974). Amino acid difference formula to help explain protein evolution. Science *185*(4154), 862–864.

Gray, V. E., Hause, R. J., Luebeck, J., Shendure, J., & Fowler, D. M. (2018). Quantitative Missense Variant Effect Prediction Using Large-Scale Mutagenesis Data. Cell Systems *6*(1), 116–124.e3.

Green, R. E., Krause, J., Briggs, A. W., Maricic, T., *et al.* (2010). A Draft Sequence of the Neandertal Genome. Science *328*(5979), 710–722.

Griesemer, D., Xue, J. R., Reilly, S. K., Ulirsch, J. C., *et al.* (2021). Genome-wide functional screen of 3'UTR variants uncovers causal variants for human disease and evolution. Cell *184*(20), 5247–5260.e19.

Groß, C., Bortoluzzi, C., de Ridder, D., Megens, H.-J., *et al.* (2020a). Prioritizing sequence variants in conserved non-coding elements in the chicken genome using chCADD. PLoS genetics *16*(9), e1009027.

Groß, C., de Ridder, D., & Reinders, M. (2018). Predicting variant deleteriousness in non-human species: applying the CADD approach in mouse. BMC Bioinformatics *19*(1), 373.

Groß, C., Derks, M., Megens, H.-J., Bosse, M., *et al.* (2020b). pCADD: SNV prioritisation in Sus scrofa. Genetics, selection, evolution: GSE *52*(1), 4.

Grüning, B., Dale, R., Sjödin, A., Chapman, B. A., *et al.* (2018). Bioconda: sustainable and comprehensive software distribution for the life sciences. Nature Methods *15*(7), 475–476.

Gulko, B., Hubisz, M. J., Gronau, I., & Siepel, A. (2015). A method for calculating probabilities of fitness consequences for point mutations across the human genome. Nature Genetics *47*(3), 276–283.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. in *Proceedings of the 34th International Conference on Machine Learning* (Precup, D. & Teh, Y. W., eds.) volume 70 of *Proceedings of Machine Learning Research*. PMLR: International Convention Centre, Sydney, Australia. 1321–1330.

Gupta, S. (2016). Annotating pathogenicity of genetic variants; a deep learning approach. Unpublished, personal communication.

Gurovich, Y., Hanani, Y., Bar, O., Nadav, G., *et al.* (2019). Identifying facial phenotypes of genetic disorders using deep learning. Nature Medicine *25*(1), 60–64.

Génin, E. (2020). Missing heritability of complex diseases: case solved?. Human Genetics *139*(1), 103–113.

Hamby, S. E., Thomas, N. S. T., Cooper, D. N., & Chuzhanova, N. (2011). A meta-analysis of single base-pair substitutions in translational termination codons ('nonstop' mutations) that cause human inherited disease. Human Genomics *5*(4), 241–264.

Harrow, J., Frankish, A., Gonzalez, J. M., Tapanari, E., *et al.* (2012). GENCODE: the reference human genome annotation for The ENCODE Project. Genome Research *22*(9), 1760–1774.

Havrilla, J. M., Pedersen, B. S., Layer, R. M., & Quinlan, A. R. (2019). A map of constrained coding regions in the human genome. Nature Genetics *51*(1), 88–95.

Heijl, S., Vroling, B., Bergh, T. v. d., & Joosten, H.-J. (2020). Mind the gap: preventing circularity in missense variant prediction. bioRxiv *2020.05.06.080424*.

Henikoff, S. & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences *89*(22), 10915–10919.

Herrero, J., Muffato, M., Beal, K., Fitzgerald, S., *et al.* (2016). Ensembl comparative genomics resources. Database *2016*, baw053.

Hirschhorn, J. N. & Daly, M. J. (2005). Genome-wide association studies for common diseases and complex traits. Nature Reviews Genetics *6*(2), 95–108.

Hoffman, M. M., Buske, O. J., Wang, J., Weng, Z., *et al.* (2012). Unsupervised pattern discovery in human chromatin structure through genomic segmentation. Nature Methods *9*(5), 473–476.

Hsieh, T.-C., Mensah, M. A., Pantel, J. T., Aguilar, D., *et al.* (2019). PEDIA: prioritization of exome data by image analysis. Genetics in Medicine *21*(12), 2807–2814.

Hu, H., Huff, C. D., Moore, B., Flygare, S., *et al.* (2013). VAAST 2.0: improved variant classification and disease-gene identification using a conservation-controlled amino acid substitution matrix. Genetic Epidemiology *37*(6), 622–634.

Hu, J. & Ng, P. C. (2012). Predicting the effects of frameshifting indels. Genome Biology *13*(2), R9.

Hu, J. & Ng, P. C. (2013). SIFT Indel: Predictions for the Functional Effects of Amino Acid Insertions/Deletions in Proteins. PLoS ONE *8*(10), e77940.

Hu, Z. & Tee, W.-W. (2017). Enhancers and chromatin structures: regulatory hubs in gene expression and diseases. Bioscience Reports *37*(2), BSR20160183.

Huang, Y.-F., Gulko, B., & Siepel, A. (2017). Fast, scalable prediction of deleterious noncoding variants from functional and population genomic data. Nature Genetics *49*(4), 618–624.

Hunt, R. C., Simhadri, V. L., Iandoli, M., Sauna, Z. E., & Kimchi-Sarfaty, C. (2014). Exposing synonymous mutations. Trends in Genetics *30*(7), 308–321.

Huynh, L. & Hormozdiari, F. (2019). TAD fusion score: discovery and ranking the contribution of deletions to genome structure. Genome Biology *20*(1), 60.

Ioannidis, N. M., Rothstein, J. H., Pejaver, V., Middha, S., *et al.* (2016). REVEL: An Ensemble Method for Predicting the Pathogenicity of Rare Missense Variants. American Journal of Human Genetics *99*(4), 877–885.

Ioffe, S. & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv *1502.03167*.

Ionita-Laza, I., McCallum, K., Xu, B., & Buxbaum, J. D. (2016). A spectral approach integrating functional genomic annotations for coding and noncoding variants. Nature Genetics *48*(2), 214–220.

Itan, Y., Shang, L., Boisson, B., Ciancanelli, M. J., *et al.* (2016). The mutation significance cutoff: gene-level thresholds for variant predictions. Nature Methods *13*(2), 109–110.

Iyama, T. & Wilson, D. M. (2013). DNA repair mechanisms in dividing and non-dividing cells. DNA Repair *12*(8), 620–636.

Jagadeesh, K. A., Paggi, J. M., Ye, J. S., Stenson, P. D., *et al.* (2019). S-CAP extends pathogenicity prediction to genetic variants that affect RNA splicing. Nature Genetics *51*(4), 755–763.

Jagadeesh, K. A., Wenger, A. M., Berger, M. J., Guturu, H., *et al.* (2016). M-CAP eliminates a majority of variants of uncertain significance in clinical exomes at high sensitivity. Nature Genetics *48*(12), 1581–1586.

Jaganathan, K., Kyriazopoulou Panagiotopoulou, S., McRae, J. F., Darbandi, S. F., *et al.* (2019). Predicting Splicing from Primary Sequence with Deep Learning. Cell *176*(3), 535–548.e24.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer: New York, NY.

Jian, X., Boerwinkle, E., & Liu, X. (2014a). In silico prediction of splice-altering single nucleotide variants in the human genome. Nucleic Acids Research *42*(22), 13534–13544.

Jian, X., Boerwinkle, E., & Liu, X. (2014b). In silico tools for splicing defect prediction: a survey from the viewpoint of end users. Genetics in Medicine *16*(7), 497–503.

Jobling, M. A. & Tyler-Smith, C. (2017). Human Y-chromosome variation in the genome-sequencing era. Nature Reviews. Genetics *18*(8), 485–497.

Jumper, J., Evans, R., Pritzel, A., Green, T., *et al.* (2021). Highly accurate protein structure prediction with AlphaFold. Nature *596*(7873), 583–589.

Jónsson, H., Sulem, P., Arnadottir, G. A., Pálsson, G., *et al.* (2018). Multiple transmissions of de novo mutations in families. Nature Genetics *50*(12), 1674–1680.

Jónsson, H., Sulem, P., Kehr, B., Kristmundsdottir, S., *et al.* (2017). Parental influence on human germline de novo mutations in 1,548 trios from Iceland. Nature *549*(7673), 519–522.

Kaplanis, J., Ide, B., Sanghvi, R., Neville, M., *et al.* (2021). Genetic and pharmacological causes of germline hypermutation. bioRxiv *2021.06.01.446180*.

Kaplanis, J., Samocha, K. E., Wiel, L., Zhang, Z., *et al.* (2020). Evidence for 28 genetic disorders discovered by combining healthcare and research data. Nature *586*(7831), 757–762.

Karczewski, K. J., Francioli, L. C., Tiao, G., Cummings, B. B., *et al.* (2020). The mutational constraint spectrum quantified from variation in 141,456 humans. Nature *581*(7809), 434–443.

Karras, T., Laine, S., Aittala, M., Hellsten, J., *et al.* (2020). Analyzing and Improving the Image Quality of StyleGAN. arXiv *1912.04958*.

Katz, Y., Wang, E. T., Airoldi, E. M., & Burge, C. B. (2010). Analysis and design of RNA sequencing experiments for identifying isoform regulation. Nature Methods *7*(12), 1009–1015.

Kaufmann, P., Pariser, A. R., & Austin, C. (2018). From scientific discovery to treatments for rare diseases – the view from the National Center for Advancing Translational Sciences – Office of Rare Diseases Research. Orphanet Journal of Rare Diseases *13*(1), 196.

Ke, G., Meng, Q., Finley, T., Wang, T., *et al.* (2017). Lightgbm: A highly efficient gradient boosting decision tree. in *Advances in neural information processing systems*. 3146–3154.

Ke, S., Shang, S., Kalachikov, S. M., Morozova, I., *et al.* (2011). Quantitative evaluation of all hexamers as exonic splicing elements. Genome Research *21*(8), 1360–1374.

Kearns, M. & Valiant, L. G. (1989). Crytographic limitations on learning Boolean formulae and finite automata. in *Proceedings of the twenty-first annual ACM symposium on Theory of computing - STOC '89*. ACM Press: Seattle, WA. 433–444.

Kelley, D. R., Snoek, J., & Rinn, J. L. (2016). Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. Genome Research *26*(7), 990–999.

Kennedy, S. R., Loeb, L. A., & Herr, A. J. (2012). Somatic mutations in aging, cancer and neurodegeneration. Mechanisms of Ageing and Development *133*(4), 118–126.

Kent, W. J., Sugnet, C. W., Furey, T. S., Roskin, K. M., *et al.* (2002). The Human Genome Browser at UCSC. Genome Research *12*(6), 996–1006.

Kent, W. J., Zweig, A. S., Barber, G., Hinrichs, A. S., & Karolchik, D. (2010). BigWig and BigBed: enabling browsing of large distributed datasets. Bioinformatics *26*(17), 2204–2207.

Khoury, G. A., Baliban, R. C., & Floudas, C. A. (2011). Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database. Scientific Reports *1*, srep00090.

Kiezun, A., Pulit, S. L., Francioli, L. C., van Dijk, F., *et al.* (2013). Deleterious Alleles in the Human Genome Are on Average Younger Than Neutral Alleles of the Same Frequency. PLoS Genetics *9*(2), e1003301.

Kircher, M., Witten, D. M., Jain, P., O'Roak, B. J., *et al.* (2014). A general framework for estimating the relative pathogenicity of human genetic variants. Nature Genetics *46*(3), 310–315.

Kircher, M., Xiong, C., Martin, B., Schubach, M., *et al.* (2019). Saturation mutagenesis of twenty disease-associated regulatory elements at single base-pair resolution. Nature Communications *10*(1), 3583.

Kleinert, P. & Kircher, M. (2021). CADD-SV – a framework to score the effects of structural variants in health and disease. bioRxiv *2021.07.10.451798*.

Kohler, J., Daneshmand, H., Lucchi, A., Zhou, M., *et al.* (2018). Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization. arXiv *1805.10694*.

Kong, A., Frigge, M. L., Masson, G., Besenbacher, S., *et al.* (2012). Rate of de novo mutations and the importance of father's age to disease risk. Nature *488*(7412), 471–475.

Korona, D. A., Lecompte, K. G., & Pursell, Z. F. (2011). The high fidelity and unique error signature of human DNA polymerase epsilon. Nucleic Acids Research *39*(5), 1763–1773.

Kruiswijk, F., Labuschagne, C. F., & Vousden, K. H. (2015). p53 in survival, death and metabolic health: a lifeguard with a licence to kill. Nature Reviews. Molecular Cell Biology *16*(7), 393–405.

Kumar, S., Harmanci, A., Vytheeswaran, J., & Gerstein, M. B. (2020). SVFX: a machine learning framework to quantify the pathogenicity of structural variants. Genome Biology *21*(1), 274.

Landrum, M. J., Lee, J. M., Benson, M., Brown, G. R., *et al.* (2018). ClinVar: improving access to variant interpretations and supporting evidence. Nucleic Acids Research *46*(D1), D1062–D1067.

Lee, D., Gorkin, D. U., Baker, M., Strober, B. J., *et al.* (2015). A method to predict the impact of regulatory variants from DNA sequence. Nature Genetics *47*(8), 955–961.

Lee, Y. & Rio, D. C. (2015). Mechanisms and Regulation of Alternative Pre-mRNA Splicing. Annual Review of Biochemistry *84*(1), 291–323.

Leslie, S., Winney, B., Hellenthal, G., Davison, D., *et al.* (2015). The fine-scale genetic structure of the British population. Nature *519*(7543), 309–314.

Lewis, B. P., Burge, C. B., & Bartel, D. P. (2005). Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are microRNA targets. Cell *120*(1), 15–20.

Lewis, C. M. & Vassos, E. (2020). Polygenic risk scores: from research tools to clinical instruments. Genome Medicine *12*(1), 44.

Li, B., Krishnan, V. G., Mort, M. E., Xin, F., *et al.* (2009). Automated inference of molecular mechanisms of disease from amino acid substitutions. Bioinformatics *25*(21), 2744–2750.

Li, H. (2011). Tabix: fast retrieval of sequence features from generic TAB-delimited files. Bioinformatics *27*(5), 718–719.

Li, S., van der Velde, K. J., de Ridder, D., van Dijk, A. D. J., *et al.* (2020). CAPICE: a computational method for Consequence-Agnostic Pathogenicity Interpretation of Clinical Exome variations. Genome Medicine *12*(1), 75.

Li, X., Kim, Y., Tsang, E. K., Davis, J. R., *et al.* (2017). The impact of rare variation on gene expression across tissues. Nature *550*(7675), 239–243.

Lieber, M. R. (2010). The mechanism of double-strand DNA break repair by the nonhomologous DNA end-joining pathway. Annual Review of Biochemistry *79*, 181–211.

Lindahl, T. (1993). Instability and decay of the primary structure of DNA. Nature *362*(6422), 709–715.

Lindblad-Toh, K., Garber, M., Zuk, O., Lin, M. F., *et al.* (2011). A high-resolution map of human evolutionary constraint using 29 mammals. Nature *478*(7370), 476–482.

Ling, J. P., Wilks, C., Charles, R., Leavey, P. J., *et al.* (2020). ASCOT identifies key regulators of neuronal subtype-specific splicing. Nature Communications *11*(1), 137.

Lipman, D. J. & Pearson, W. R. (1985). Rapid and sensitive protein similarity searches. Science *227*(4693), 1435–1441.

Liu, X., Wu, C., Li, C., & Boerwinkle, E. (2016). dbNSFP v3.0: A One-Stop Database of Functional Predictions and Annotations for Human Nonsynonymous and Splice-Site SNVs. Human Mutation *37*(3), 235–241.

Liu, Y., Yang, Q., & Zhao, F. (2021). Synonymous but Not Silent: The Codon Usage Code for Gene Expression and Protein Folding. Annual Review of Biochemistry *90*(1), 375–401.

Livingstone, M., Folkman, L., Yang, Y., Zhang, P., *et al.* (2017). Investigating DNA-, RNA-, and protein-based features as a means to discriminate pathogenic synonymous variants. Human Mutation *38*(10), 1336–1347.

Loewenstein, Y., Raimondo, D., Redfern, O. C., Watson, J., *et al.* (2009). Protein function annotation by homology-based inference. Genome Biology *10*(2), 207.

Logsdon, G. A., Vollger, M. R., & Eichler, E. E. (2020). Long-read human genome sequencing and its applications. Nature Reviews. Genetics *21*(10), 597–614.

Lohse, K. & Frantz, L. A. F. (2014). Neandertal Admixture in Eurasia Confirmed by Maximum-Likelihood Analysis of Three Genomes. Genetics *196*(4), 1241–1251.

Low, Y., Gonzalez, J. E., Kyrola, A., Bickson, D., *et al.* (2014). GraphLab: A New Framework For Parallel Machine Learning. arXiv *1408.2041*.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., *et al.* (2020). From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence *2*(1), 56–67.

Lupski, J. R., Belmont, J. W., Boerwinkle, E., & Gibbs, R. A. (2011). Clan genomics and the complex architecture of human disease. Cell *147*(1), 32–43.

MacArthur, D. G., Manolio, T. A., Dimmock, D. P., Rehm, H. L., *et al.* (2014). Guidelines for investigating causality of sequence variants in human disease. Nature *508*(7497), 469–476.

Maki, H. (2002). Origins of Spontaneous Mutations: Specificity and Directionality of Base-Substitution, Frameshift, and Sequence-Substitution Mutageneses. Annual Review of Genetics *36*(1), 279–303.

Makrythanasis, P. & Antonarakis, S. E. (2013). Pathogenic variants in non-protein-coding sequences. Clinical Genetics *84*(5), 422–428.

Manolio, T. A., Collins, F. S., Cox, N. J., Goldstein, D. B., *et al.* (2009). Finding the missing heritability of complex diseases. Nature *461*(7265), 747–753.

Mao, P. & Wyrick, J. J. (2019). Organization of DNA damage, excision repair, and mutagenesis in chromatin: A genomic perspective. DNA repair *81*, 102645.

Marouli, E., Graff, M., Medina-Gomez, C., Lo, K. S., *et al.* (2017). Rare and low-frequency coding variants alter human adult height. Nature *542*(7640), 186–190.

Marshall, C. R., Chowdhury, S., Taft, R. J., Lebo, M. S., *et al.* (2020). Best practices for the analytical validation of clinical whole-genome sequencing intended for the diagnosis of germline disease. npj Genomic Medicine *5*(1), 1–12.

Marth, G. T., Yu, F., Indap, A. R., Garimella, K., *et al.* (2011). The functional spectrum of low-frequency coding variation. Genome Biology *12*(9), R84.

Matalonga, L., Hernández-Ferrer, C., Piscia, D., Schüle, R., *et al.* (2021). Solving patients with rare diseases through programmatic reanalysis of genome-phenome data. European Journal of Human Genetics *29*(9), 1337–1347.

Mather, C. A., Mooney, S. D., Salipante, S. J., Scroggins, S., *et al.* (2016). CADD score has limited clinical validity for the identification of pathogenic variants in noncoding regions in a hereditary cancer panel. Genetics in Medicine *18*(12), 1269–1275.

Matreyek, K. A., Starita, L. M., Stephany, J. J., Martin, B., *et al.* (2018). Multiplex assessment of protein variant abundance by massively parallel sequencing. Nature Genetics *50*(6), 874–882.

Mazzucato, M., Visonà Dalla Pozza, L., Manea, S., Minichiello, C., & Facchin, P. (2014). A population-based registry as a source of health indicators for rare diseases: the ten-year experience of the Veneto Region's rare diseases registry. Orphanet Journal of Rare Diseases *9*(1), 37.

McCormick, E. M., Lott, M. T., Dulik, M. C., Shen, L., *et al.* (2020). Specifications of the ACMG/AMP standards and guidelines for mitochondrial DNA variant interpretation. Human Mutation *41*(12), 2028–2057.

McFarland, C. D., Korolev, K. S., Kryukov, G. V., Sunyaev, S. R., & Mirny, L. A. (2013). Impact of deleterious passenger mutations on cancer progression. Proceedings of the National Academy of Sciences *110*(8), 2910–2915.

McLaren, W., Gil, L., Hunt, S. E., Riat, H. S., *et al.* (2016). The Ensembl Variant Effect Predictor. Genome Biology *17*(1), 122.

McLaughlin, R. N., Poelwijk, F. J., Raman, A., Gosal, W. S., & Ranganathan, R. (2012). The spatial architecture of protein function and adaptation. Nature *491*(7422), 138–142.

McVicker, G., Gordon, D., Davis, C., & Green, P. (2009). Widespread Genomic Signatures of Natural Selection in Hominid Evolution. PLoS Genetics *5*(5), e1000471.

Michaelson, J. J., Shi, Y., Gujral, M., Zheng, H., *et al.* (2012). Whole-genome sequencing in autism identifies hot spots for de novo germline mutation. Cell *151*(7), 1431–1442.

Mitchell, P. J. & Tjian, R. (1989). Transcriptional regulation in mammalian cells by sequence-specific DNA binding proteins. Science *245*(4916), 371–378.

Mitelman, F., Johansson, B., & Mertens, F. (2007). The impact of translocations and gene fusions on cancer causation. Nature Reviews Cancer *7*(4), 233–245.

Mohammadi, P., Castel, S. E., Cummings, B. B., Einson, J., *et al.* (2019). Genetic regulatory variation in populations informs transcriptome analysis in rare disease. Science *366*(6463), 351–356.

Morjaria, S. (2021). Driver mutations in oncogenesis. International Journal of Molecular & Immuno Oncology *6*, 100–102.

Mort, M., Sterne-Weiler, T., Li, B., Ball, E. V., *et al.* (2014). MutPred Splice: machine learning-based prediction of exonic variants that disrupt splicing. Genome Biology *15*(1), R19.

Mostafavi, H., Harpak, A., Agarwal, I., Conley, D., *et al.* (2020). Variable prediction accuracy of polygenic scores within an ancestry group. eLife *9*, e48376.

Muerdter, F., Boryn, L. M., & Arnold, C. D. (2015). STARR-seq — Principles and applications. Genomics *106*(3), 145–150.

Mukamel, R. E., Handsaker, R. E., Sherman, M. A., Barton, A. R., *et al.* (2021). Protein-coding repeat polymorphisms strongly shape diverse human phenotypes. Science *373*(6562), 1499–1505.

Mundhenk, T. N., Chen, B. Y., & Friedland, G. (2020). Efficient Saliency Maps for Explainable AI. arXiv *1911.11293*.

Mölder, F., Jablonski, K. P., Letcher, B., Hall, M. B., *et al.* (2021). Sustainable data analysis with Snakemake. F1000Research *10*, 33.

Nagy, E. & Maquat, L. E. (1998). A rule for termination-codon position within intron-containing genes: when nonsense affects RNA abundance. Trends in Biochemical Sciences *23*(6), 198–199.

Need, A. C., Shashi, V., Hitomi, Y., Schoch, K., *et al.* (2012). Clinical application of exome sequencing in undiagnosed genetic conditions. Journal of Medical Genetics *49*(6), 353–361.

Ng, P. C. & Henikoff, S. (2003). SIFT: Predicting amino acid changes that affect protein function. Nucleic Acids Research *31*(13), 3812–3814.

Ng, S. B., Turner, E. H., Robertson, P. D., Flygare, S. D., *et al.* (2009). Targeted capture and massively parallel sequencing of 12 human exomes. Nature *461*(7261), 272–276.

Nurk, S., Koren, S., Rhie, A., Rautiainen, M., *et al.* (2021). The complete sequence of a human genome. bioRxiv *2021.05.26.445798*.

Nykamp, K., Anderson, M., Powers, M., Garcia, J., *et al.* (2017). Sherloc: a comprehensive refinement of the ACMG-AMP variant classification criteria. Genetics in Medicine: Official Journal of the American College of Medical Genetics *19*(10), 1105–1117.

Ostrander, E. A., Davis, B. W., & Ostrander, G. K. (2016). Transmissible Tumors: Breaking the Cancer Paradigm. Trends in Genetics *32*(1), 1–15.

Park, E., Pan, Z., Zhang, Z., Lin, L., & Xing, Y. (2018). The Expanding Landscape of Alternative Splicing Variation in Human Populations. American Journal of Human Genetics *102*(1), 11–26.

Patwardhan, R. P., Hiatt, J. B., Witten, D. M., Kim, M. J., *et al.* (2012). Massively parallel functional dissection of mammalian enhancers in vivo. Nature Biotechnology *30*(3), 265–270.

Patwardhan, R. P., Lee, C., Litvin, O., Young, D. L., *et al.* (2009). High-resolution analysis of DNA regulatory elements by synthetic saturation mutagenesis. Nature Biotechnology *27*(12), 1173–1175.

Payne, A., Holmes, N., Rakyan, V., & Loose, M. (2019). BulkVis: a graphical viewer for Oxford nanopore bulk FAST5 files. Bioinformatics *35*(13), 2193–2198.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., *et al.* (2011). Scikit-learn: Machine learning in python. Journal of Machine Learning Research *12*(85), 2825–2830.

Peng, R. D. (2011). Reproducible Research in Computational Science. Science *334*(6060), 1226–1227.

Phan, L., Jin, Y., Zhang, H., Qiang, W., *et al.* (2020). Alfa: Allele frequency aggregator. https://www.ncbi.nlm.nih.gov/snp/docs/gsr/alfa/.

Phillips, D. H. (2018). Mutational spectra and mutational signatures: Insights into cancer aetiology and mechanisms of DNA damage and repair. DNA Repair *71*, 6–11.

Pogue, R. E., Cavalcanti, D. P., Shanker, S., Andrade, R. V., *et al.* (2018). Rare genetic diseases: update on diagnosis, treatment and online resources. Drug Discovery Today *23*(1), 187–195.

Pollard, K. S., Hubisz, M. J., Rosenbloom, K. R., & Siepel, A. (2010). Detection of nonneutral substitution rates on mammalian phylogenies. Genome Research *20*(1), 110–121.

Ponting, C. P. (2017). Biological function in the twilight zone of sequence conservation. BMC Biology *15*(1), 71.

Posey, J. E., O'Donnell-Luria, A. H., Chong, J. X., Harel, T., *et al.* (2019). Insights into genetics, human biology and disease gleaned from family based genomic studies. Genetics in Medicine *21*(4), 798–812.

Poznik, G. D., Henn, B. M., Yee, M.-C., Sliwerska, E., *et al.* (2013). Sequencing Y Chromosomes Resolves Discrepancy in Time to Common Ancestor of Males Versus Females. Science *341*(6145), 562–565.

Pruitt, K. D. (2004). NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Research *33*(D1), D501–D504.

Quang, D., Chen, Y., & Xie, X. (2015). DANN: a deep learning approach for annotating the pathogenicity of genetic variants. Bioinformatics *31*(5), 761–763.

Ramensky, V., Bork, P., & Sunyaev, S. (2002). Human non-synonymous SNPs: server and survey. Nucleic Acids Research *30*(17), 3894–3900.

Rands, C. M., Meader, S., Ponting, C. P., & Lunter, G. (2014). 8.2% of the Human genome is constrained: variation in rates of turnover across functional element classes in the human lineage. PLoS genetics *10*(7), e1004525.

Raney, B. J., Dreszer, T. R., Barber, G. P., Clawson, H., *et al.* (2014). Track data hubs enable visualization of user-defined genome-wide annotations on the UCSC Genome Browser. Bioinformatics *30*(7), 1003–1005.

Rausell, A., Luo, Y., Lopez, M., Seeleuthner, Y., *et al.* (2020). Common homozygosity for predicted loss-of-function variants reveals both redundant and advantageous effects of dispensable human genes. Proceedings of the National Academy of Sciences *117*(24), 13626–13636.

Rees, D. C., Williams, T. N., & Gladwin, M. T. (2010). Sickle-cell disease. The Lancet *376*(9757), 2018–2031.

Rees, J. S., Castellano, S., & Andrés, A. M. (2020). The Genomics of Human Local Adaptation. Trends in Genetics *36*(6), 415–428.

Reese, M. G., Eeckman, F. H., Kulp, D., & Haussler, D. (1997). Improved Splice Site Detection in Genie. Journal of Computational Biology *4*(3), 311–323.

Reese, M. G., Moore, B., Batchelor, C., Salas, F., *et al.* (2010). A standard variation file format for human genome sequences. Genome Biology *11*(8), R88.

Rehm, H. L., Bale, S. J., Bayrak-Toydemir, P., Berg, J. S., *et al.* (2013). ACMG clinical laboratory standards for next-generation sequencing. Genetics in Medicine *15*(9), 733–747.

Rehm, H. L., Berg, J. S., Brooks, L. D., Bustamante, C. D., *et al.* (2015). ClinGen — The Clinical Genome Resource. New England Journal of Medicine *372*(23), 2235–2242.

Reich, D., Green, R. E., Kircher, M., Krause, J., *et al.* (2010). Genetic history of an archaic hominin group from Denisova Cave in Siberia. Nature *468*(7327), 1053–1060.

Rentzsch, P., Schubach, M., Shendure, J., & Kircher, M. (2021). CADD-Splice — improving genome-wide variant effect prediction using deep learning-derived splice scores. Genome Medicine *13*(1), 31.

Rentzsch, P., Witten, D., Cooper, G. M., Shendure, J., & Kircher, M. (2019). CADD: predicting the deleteriousness of variants throughout the human genome. Nucleic Acids Research *47*(D1), D886–D894.

Reva, B., Antipin, Y., & Sander, C. (2011). Predicting the functional impact of protein mutations: application to cancer genomics. Nucleic Acids Research *39*(17), e118–e118.

Richards, S., Aziz, N., Bale, S., Bick, D., *et al.* (2015). Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology. Genetics in Medicine *17*(5), 405–423.

Ripley, B. D. (2009). Pattern recognition and neural networks. Cambridge Univ. Press: Cambridge, UK.

Ritchie, G. R. S., Dunham, I., Zeggini, E., & Flicek, P. (2014). Functional annotation of non-coding sequence variants. Nature methods *11*(3), 294–296.

Rivas, M. A., Pirinen, M., Conrad, D. F., Lek, M., *et al.* (2015). Human genomics. Effect of predicted protein-truncating genetic variants on the human transcriptome. Science *348*(6235), 666–669.

Roberts, S. A. & Gordenin, D. A. (2014). Hypermutation in human cancer genomes: footprints and mechanisms. Nature Reviews. Cancer *14*(12), 786–800.

Robinson, J. T., Thorvaldsdóttir, H., Winckler, W., Guttman, M., *et al.* (2011). Integrative genomics viewer. Nature Biotechnology *29*(1), 24–26.

Robinson, P. N., Köhler, S., Bauer, S., Seelow, D., *et al.* (2008). The Human Phenotype Ontology: a tool for annotating and analyzing human hereditary disease. American Journal of Human Genetics *83*(5), 610–615.

Rodenburg, R. J. (2018). The functional genomics laboratory: functional validation of genetic variants. Journal of Inherited Metabolic Disease *41*(3), 297–307.

Rogers, M. F., Shihab, H. A., Mort, M., Cooper, D. N., *et al.* (2018). FATHMM-XF: accurate prediction of pathogenic point mutations via extended features. Bioinformatics *34*(3), 511–513.

Rogozin, I. B. & Pavlov, Y. I. (2003). Theoretical analysis of mutation hotspots and their DNA sequence context specificity. Mutation Research *544*(1), 65–85.

Rosenberg, A., Patwardhan, R., Shendure, J., & Seelig, G. (2015). Learning the Sequence Determinants of Alternative Splicing from Millions of Random Sequences. Cell *163*(3), 698–711.

Ruffier, M., Kähäri, A., Komorowska, M., Keenan, S., *et al.* (2017). Ensembl core software resources: storage and programmatic access for DNA sequence and genome annotation. Database *2017*, bax020.

Sabarinathan, R., Mularoni, L., Deu-Pons, J., Gonzalez-Perez, A., & López-Bigas, N. (2016). Nucleotide excision repair is impaired by binding of transcription factors to DNA. Nature *532*(7598), 264–267.

Sample, P. J., Wang, B., Reid, D. W., Presnyak, V., *et al.* (2019). Human 5' UTR design and variant effect prediction from a massively parallel translation assay. Nature Biotechnology *37*(7), 803–809.

Sanchis-Juan, A., Stephens, J., French, C. E., Gleadall, N., *et al.* (2018). Complex structural variants in Mendelian disorders: identification and breakpoint resolution using short- and long-read genome sequencing. Genome Medicine *10*(1), 95.

Sandelin, A., Alkema, W., Engström, P., Wasserman, W. W., & Lenhard, B. (2004). JASPAR: an open-access database for eukaryotic transcription factor binding profiles. Nucleic Acids Research *32*(D1), D91–D94.

Sayers, E. W., Barrett, T., Benson, D. A., Bolton, E., *et al.* (2011). Database resources of the National Center for Biotechnology Information. Nucleic Acids Research *39*(D1), D38–D51.

Scally, A., Dutheil, J. Y., Hillier, L. W., Jordan, G. E., *et al.* (2012). Insights into hominid evolution from the gorilla genome sequence. Nature *483*(7388), 169–175.

Schaid, D. J., Chen, W., & Larson, N. B. (2018). From genome-wide associations to candidate causal variants by statistical fine-mapping. Nature Reviews Genetics *19*(8), 491–504.

Schmidt, F., Kern, F., & Schulz, M. H. (2020). Integrative prediction of gene expression with chromatin accessibility and conformation data. Epigenetics & Chromatin *13*(1), 4.

Schmitt, M. W., Matsumoto, Y., & Loeb, L. A. (2009). High fidelity and lesion bypass capability of human DNA polymerase delta. Biochimie *91*(9), 1163–1172.

Schneider, V. A., Graves-Lindsay, T., Howe, K., Bouk, N., *et al.* (2017). Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. Genome Research *27*(5), 849–864.

Schwarz, J. M., Cooper, D. N., Schuelke, M., & Seelow, D. (2014). MutationTaster2: mutation prediction for the deep-sequencing age. Nature Methods *11*(4), 361–362.

Sharma, R., Lewis, S., & Wlodarski, M. W. (2020). DNA Repair Syndromes and Cancer: Insights Into Genetics and Phenotype Patterns. Frontiers in Pediatrics *8*, 570084.

Shen, S., Park, J. W., Huang, J., Dittmar, K. A., *et al.* (2012). MATS: a Bayesian framework for flexible detection of differential alternative splicing from RNA-Seq data. Nucleic Acids Research *40*(8), e61–e61.

Shihab, H. A., Gough, J., Mort, M., Cooper, D. N., *et al.* (2014). Ranking non-synonymous single nucleotide polymorphisms based on disease concepts. Human Genomics *8*, 11.

Siepel, A., Bejerano, G., Pedersen, J. S., Hinrichs, A. S., *et al.* (2005). Evolutionarily conserved elements in vertebrate, insect, worm, and yeast genomes. Genome Research *15*(8), 1034–1050.

Simpson, J. T. & Pop, M. (2015). The Theory and Practice of Genome Sequence Assembly. Annual Review of Genomics and Human Genetics *16*(1), 153–172.

Slatkin, M. (2008). Linkage disequilibrium — understanding the evolutionary past and mapping the medical future. Nature Reviews Genetics *9*(6), 477–485.

Smedley, D., Schubach, M., Jacobsen, J., Köhler, S., *et al.* (2016). A Whole-Genome Analysis Framework for Effective Identification of Pathogenic Regulatory Variants in Mendelian Disease. American Journal of Human Genetics *99*(3), 595–606.

Soemedi, R., Cygan, K. J., Rhine, C. L., Wang, J., *et al.* (2017). Pathogenic variants that alter protein code often disrupt splicing. Nature Genetics *49*(6), 848–855.

Starita, L. M., Ahituv, N., Dunham, M. J., Kitzman, J. O., *et al.* (2017). Variant Interpretation: Functional Assays to the Rescue. American Journal of Human Genetics *101*(3), 315–325.

Starita, L. M., Young, D. L., Islam, M., Kitzman, J. O., *et al.* (2015). Massively Parallel Functional Analysis of BRCA1 RING Domain Variants. Genetics *200*(2), 413–422.

Stark, Z., Tan, T. Y., Chong, B., Brett, G. R., *et al.* (2016). A prospective evaluation of whole-exome sequencing as a first-tier molecular test in infants with suspected monogenic disorders. Genetics in Medicine *18*(11), 1090–1096.

Stein, L. (2001). Genome annotation: from sequence to biology. Nature Reviews. Genetics *2*(7), 493–503.

Stenson, P. D., Mort, M., Ball, E. V., Chapman, M., *et al.* (2020). The Human Gene Mutation Database (HGMD®): optimizing its use in a clinical diagnostic or research setting. Human Genetics *139*(10), 1197–1207.

Stevenson, D. A. & Carey, J. C. (2004). Contribution of malformations and genetic disorders to mortality in a children's hospital. American Journal of Medical Genetics *126A*(4), 393–397.

Stoeger, T., Gerlach, M., Morimoto, R. I., & Amaral, L. A. N. (2018). Large-scale investigation of the reasons why potentially important genes are ignored. PLOS Biology *16*(9), e2006643.

Stratton, M. R., Campbell, P. J., & Futreal, P. A. (2009). The cancer genome. Nature *458*(7239), 719–724.

Su, J., Vargas, D. V., & Kouichi, S. (2019). One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation *23*(5), 828–841.

Sudmant, P. H., Rausch, T., Gardner, E. J., Handsaker, R. E., *et al.* (2015). An integrated map of structural variation in 2,504 human genomes. Nature *526*(7571), 75–81.

Sundaram, L., Gao, H., Padigepati, S. R., McRae, J. F., *et al.* (2018). Predicting the clinical impact of human mutation with deep neural networks. Nature Genetics *50*(8), 1161–1170.

Sunyaev, S., Ramensky, V., Koch, I., Lathe, W., *et al.* (2001). Prediction of deleterious human alleles. Human Molecular Genetics *10*(6), 591–597.

Szustakowski, J. D., Balasubramanian, S., Sasson, A., Khalid, S., *et al.* (2020). Advancing Human Genetics Research and Drug Discovery through Exome Sequencing of the UK Biobank. medrXiv *2020.11.02.20222232*.

Ségurel, L., Wyman, M. J., & Przeworski, M. (2014). Determinants of Mutation Rate Variation in the Human Germline. Annual Review of Genomics and Human Genetics *15*(1), 47–70.

Taliun, D., Harris, D. N., Kessler, M. D., Carlson, J., *et al.* (2021). Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program. Nature *590*(7845), 290–299.

Tam, V., Patel, N., Turcotte, M., Bossé, Y., *et al.* (2019). Benefits and limitations of genome-wide association studies. Nature Reviews Genetics *20*(8), 467–484.

Taruscio, D., Groft, S. C., Cederroth, H., Melegh, B., *et al.* (2015). Undiagnosed Diseases Network International (UDNI): White paper for global actions to meet patient needs. Molecular Genetics and Metabolism *116*(4), 223–225.

Tate, J. G., Bamford, S., Jubb, H. C., Sondka, Z., *et al.* (2019). COSMIC: the Catalogue Of Somatic Mutations In Cancer. Nucleic Acids Research *47*(D1), D941–D947.

The 1000 Genomes Project Consortium (2011). Variation in genome-wide mutation rates within and between human families. Nature Genetics *43*(7), 712–714.

The 1000 Genomes Project Consortium (2015). A global reference for human genetic variation. Nature *526*(7571), 68–74.

Thomas, P. D., Kejariwal, A., Campbell, M. J., Mi, H., *et al.* (2003). PANTHER: a browsable database of gene products organized by biological function, using curated protein family and subfamily classification. Nucleic Acids Research *31*(1), 334–341.

Ule, J. & Blencowe, B. J. (2019). Alternative Splicing Regulatory Networks: Functions, Mechanisms, and Evolution. Molecular Cell *76*(2), 329–345.

Vihinen, M. (2012). How to evaluate performance of prediction methods? Measures and their interpretation in variation effect analysis. BMC Genomics *13 Suppl 4*, S2.

Visscher, P. M., Wray, N. R., Zhang, Q., Sklar, P., *et al.* (2017). 10 Years of GWAS Discovery: Biology, Function, and Translation. American Journal of Human Genetics *101*(1), 5–22.

von Mering, C., Jensen, L. J., Snel, B., Hooper, S. D., *et al.* (2005). STRING: known and predicted protein-protein associations, integrated and transferred across organisms. Nucleic Acids Research *33*(D1), D433–D437.

Wagih, O., Galardini, M., Busby, B. P., Memon, D., *et al.* (2018). A resource of variant effect predictions of single nucleotide variants in model organisms. Molecular Systems Biology *14*(12).

Wang, K., Li, M., & Hakonarson, H. (2010). ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. Nucleic Acids Research *38*(16), e164–e164.

Wang, X., Zhou, T., Wunderlich, Z., Maurano, M. T., *et al.* (2018). Analysis of Genetic Variation Indicates DNA Shape Involvement in Purifying Selection. Molecular Biology and Evolution *35*(8), 1958–1967.

Whiffin, N., Karczewski, K. J., Zhang, X., Chothani, S., *et al.* (2020). Characterising the loss-of-function impact of 5' untranslated region variants in 15,708 individuals. Nature Communications *11*(1), 2523.

Wright, C. F., Fitzgerald, T. W., Jones, W. D., Clayton, S., *et al.* (2015). Genetic diagnosis of developmental disorders in the DDD study: a scalable analysis of genome-wide research data. The Lancet *385*(9975), 1305–1314.

Wright, C. F., McRae, J. F., Clayton, S., Gallone, G., *et al.* (2018). Making new genetic diagnoses with old data: iterative reanalysis and reporting from genome-wide data in 1,133 families with developmental disorders. Genetics in Medicine: Official Journal of the American College of Medical Genetics *20*(10), 1216–1223.

wwPDB consortium (2019). Protein Data Bank: the single global archive for 3D macromolecular structure data. Nucleic Acids Research *47*(D1), D520–D528.

Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., *et al.* (2015). The human splicing code reveals new insights into the genetic determinants of disease. Science *347*(6218), 1254806.

Yandell, M., Huff, C., Hu, H., Singleton, M., *et al.* (2011). A probabilistic disease-gene finder for personal genomes. Genome Research *21*(9), 1529–1542.

Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., & Schoenholz, S. S. (2019). A Mean Field Theory of Batch Normalization. arXiv *1902.08129*.

Yang, J., Jin, Z.-B., Chen, J., Huang, X.-F., *et al.* (2017a). Genetic signatures of high-altitude adaptation in Tibetans. Proceedings of the National Academy of Sciences *114*(16), 4189–4194.

Yang, M. & Kim, B. (2019). Benchmarking Attribution Methods with Relative Feature Importance. arXiv *1907.09701*.

Yang, S., Lincoln, S. E., Kobayashi, Y., Nykamp, K., *et al.* (2017b). Sources of discordance among germ-line variant classifications in ClinVar. Genetics in Medicine *19*(10), 1118–1126.

Yeo, G. & Burge, C. B. (2004). Maximum Entropy Modeling of Short Sequence Motifs with Applications to RNA Splicing Signals. Journal of Computational Biology *11*(2-3), 377–394.

Yizhak, K., Aguet, F., Kim, J., Hess, J. M., *et al.* (2019). RNA sequence analysis reveals macroscopic somatic clonal expansion across normal tissues. Science *364*(6444).

Yu, D.-H., Waterland, R. A., Zhang, P., Schady, D., *et al.* (2014). Targeted p16Ink4a epimutation causes tumorigenesis and reduces survival in mice. Journal of Clinical Investigation *124*(9), 3708–3712.

Zeitlinger, J. (2020). Seven myths of how transcription factors read the cis-regulatory code. Current Opinion in Systems Biology *23*, 22–31.

Zerbino, D. R., Wilder, S. P., Johnson, N., Juettemann, T., & Flicek, P. R. (2015). The Ensembl Regulatory Build. Genome Biology *16*(1), 56.

Zhang, N., Huang, T., & Cai, Y.-D. (2015). Discriminating between deleterious and neutral non-frameshifting indels based on protein interaction networks and hybrid properties. Molecular Genetics and Genomics *290*(1), 343–352.

Zhang, P., Bigio, B., Rapaport, F., Zhang, S.-Y., *et al.* (2018). PopViz: a webserver for visualizing minor allele frequencies and damage prediction scores of human genetic variations. Bioinformatics *34*(24), 4307–4309.

Zhang, X., Lin, H., Zhao, H., Hao, Y., *et al.* (2014). Impact of human pathogenic micro-insertions and micro-deletions on post-transcriptional regulation. Human Molecular Genetics *23*(11), 3024–3034.

Zhao, H., Sun, Z., Wang, J., Huang, H., *et al.* (2014). CrossMap: a versatile tool for coordinate conversion between genome assemblies. Bioinformatics *30*(7), 1006–1007.

Zhao, H., Yang, Y., Lin, H., Zhang, X., *et al.* (2013). DDIG-in: discriminating between disease-associated and neutral non-frameshifting micro-indels. Genome Biology *14*(3), R23.

Zhou, J. & Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. Nature Methods *12*(10), 931–934.

Zhou, T., Yang, L., Lu, Y., Dror, I., *et al.* (2013). DNAshape: a method for the high-throughput prediction of DNA structural features on a genomic scale. Nucleic Acids Research *41*(W1), W56–W62.

Zhu, X., Petrovski, S., Xie, P., Ruzzo, E. K., *et al.* (2015). Whole-exome sequencing in undiagnosed genetic diseases: interpreting 119 trios. Genetics in Medicine: Official Journal of the American College of Medical Genetics *17*(10), 774–781.

Zia, A. & Moses, A. M. (2011). Ranking insertion, deletion and nonsense mutations based on their effect on genetic information. BMC Bioinformatics *12*, 299.

Zook, J. M., McDaniel, J., Olson, N. D., Wagner, J., *et al.* (2019). An open resource for accurately benchmarking small variant and reference calls. Nature Biotechnology *37*(5), 561–566.

List of Figures

List of Tables

List of Sample codes

Glossary

accuracy rate of getting the correct result in a sequence

allele one of multiple versions of a gene

allele frequency frequency that an allele different from the reference appears in the total population. If individuals have on average one copy: $AF = 0.5$

API Application Programming Interface, enables interactions between computer programs

AUC Area Under the Curve

auPRC area under the Precision-Recall Curve

auROC area under the Receiver Operator Characteristic

bp base pair, matching nucleotides on each of the two DNA strands

CADD "Combined Annotation - Dependent Depletion" - variant classifier that is main topic of this thesis

coding protein encoding region, region of the genome that gets translated into proteins

conservation metric that describes the stability of genome segments in terms of mutations on evolutionary time scales

contig contiguous stretch of DNA sequence without gaps in a reference genome

DMS Deep Mutational Scan, assay that evaluates all possible amino acid substitutions in a sequence

Ensembl genome data project by the European Bioinformatics Institute

exon part of a gene that encodes the final mRNA

GWAS Genome-Wide Association Study

gene segment of the genome that encodes the blue print for one or more functional products

genome entirety of an organisms genetic information (void of duplications)

genotype genetic information of an organism

haplotype one half of a diploid chromosome

human bipedal creature that tries to understand things, very common on earth

InDel Insertion andor Deletion, usually shorter than 50 bp to distinguish from SV

intron part of a gene that get excluded from the mRNA by splicing

kb kilo base, a sequence of a thousand nucleotide bases

liftover genome coordinate transfer between reference builds

MAF Minor Allele Frequency, frequency of second most common allele in a population

MAVE Multiplexed Assays of Variant Effect - reporter assay that measures a biological readout for many variants

MFASS Multiplexed Functional Assay of Splicing using Sort-seq

mutation directed change in a nucleotide sequence

NCBI National Center for Biotechnology Information

nucleotide sub-unit of DNA and RNA, containing a five-carbon sugar molecule, a nucleobase and a phosphate group

one-hot-encoding transcribing a categorical term in one binary variable per category

pathogenicity potential to cause a disease

phasing determining if variants are on the same allele/haplotype

phenotype observable trait of an organism

Phred negative log 10 scaling of ranks, named after a software

PRC Precision-Recall Curve, metric to benchmark models on distinguishing two sets, dependent on size of the sets

ROC Receiver Operator Characteristic, metric to benchmark models on distinguishing two sets, independent of size of the sets

SNV Single Nucleotide Variant, mutation that substitutes a single base pair in a DNA sequence by another

SV Structural Variant, large scale difference in a DNA sequence

TF Transcription Factor, DNA binding molecule whose binding influences the mRNA level

transcription copying of genetic information from DNA to RNA

translation protein synthesis by conversion of information from nucleotides (mRNA) to amino acids

variant difference between two nucleotide sequences

VEP Variant Effect Predictor, software for annotating genetic variants

Eigenständigkeitserklärung

Erklärung gemäß §7 Abs. 5 b (Monographie) nach Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät vom 11.07.2018

Name, Vorname: Rentzsch, Philipp

Promotionsfach: Informatik

Thema der Dissertation: Using machine learning to predict pathogenicity of genomic variants throughout the human Genome

Aufstellung der Publikationen

1. Rentzsch, P., Witten, D., Cooper, G. M., Shendure, J., & Kircher, M. (2019). CADD: predicting the deleteriousness of variants throughout the human genome. Nucleic Acids Research 47(D1), D886–D894.
2. Rentzsch, P., Schubach, M., Shendure, J., & Kircher, M. (2021). CADD-Splice - improving genome-wide variant effect prediction using deep learning-derived splice scores. Genome Medicine 13(1), 31.

Darlegung des eigenen Anteils der oben genannten Publikationen

Die Kapitel 5, 6 und 7 dieser Dissertation basieren inhaltlich auf den oben genannten Publikationen. Alle präsentierten Analysen, unabhängig davon ob aus den Publikationen stammend oder nur hier gezeigt, wurden von mir allein erstellt. Meine Koautoren Martin Kircher, Jay Shendure, Daniela Witten, Greg Cooper und Max Schubach haben zu den daraus folgenden Texten beigetragen, wie in den entsprechenden Abschnitten der Publikationen dargelegt. Alle Texte wurden für diese Arbeit grundlegend durch mich überarbeitet, neu strukturiert, aktualisiert und ergänzt. Nichtsdestotrotz können Sätze und Formulierungen in Teilen mit den Publikationen übereinstimmen. Die Abbildungen der Kapitel 5, 6 und 7 überlappen inhaltlich ebenfalls mit den beiden Publikation, sind jedoch alle durch mich erstellt und überarbeitet worden.

Berlin, 12.11.2021

Philipp Rentzsch

Zustimmung des Betreuers erfolgt nach aktueller Lage per E-Mail