Prop3D: A flexible, Python-based platform for machine learning with protein structural properties and biophysical data

Eli J. Draizen[1,2*], Luis Felipe R. Murillo[3], John Readey[4], Cameron Mura[2*], Philip E. Bourne[1,2*]

1 Department of Biomedical Engineering; University of Virginia; Charlottesville, Virginia; USA
2 School of Data Science; University of Virginia; Charlottesville, Virginia; USA
3 Department of Anthropology; University of Notre Dame; Notre Dame, Indiana; USA
4 The HDF Group; Bellevue, Washington; USA

*Correspondence: ed4bu@virginia.edu, cmura@virginia.edu, peb6a@virginia.edu

Contents

1 Introduction **2**

2 Motivating Factors: Data Leakage, Biophysical Properties, & Protein Representations **4**
2.1 Evolutionary Data Leakage . . . 4
2.2 Biophysical Properties in ML . . . 5
2.3 Protein Representations . . . 6
2.3.1 Proteins in 1D: Sequences . . . 6
2.3.2 Proteins in 2D: Residue-residue Graphs . . . 7
2.3.3 Proteins in 3D: Structures as 3D Volumetric Data . . . 7
2.4 Outline of This Work . . . 8

3 Overview of the Software & Dataset **8**
3.1 Architecture and Design . . . 8
3.2 Meadowlark: An Extensible, Dockerized Toolkit for Reproducible, Cross-platform Structural Bioinformatics Workflows . . . 9
3.3 AtomicToil: Mapping Structural Information to Sets of Massively Parallel Tasks . . . 10
3.4 Capabilities and Features . . . 10
3.4.1 Protein Structure Preparation . . . 10
3.4.2 Biophysical Property Calculation & Featurization . . . 11
3.5 Dataset Design and Open Data Format . . . 11

4 Data Availability & Prop3D's FAIRness **14**

5 Summary & Outlook **16**

Abstract

Machine learning (ML) has a rich history in structural bioinformatics, and modern approaches, such as deep learning, are revolutionizing our knowledge of the subtle

relationships between biomolecular sequence, structure, function, dynamics and evolution. As with any advance that rests upon statistical learning approaches, the recent progress in biomolecular sciences is enabled by the availability of vast volumes of sufficiently-variable data. To be of utility, such datasets must be well-structured, machine-readable, intelligible and manipulable. These and related requirements pose challenges that become especially acute at the computational scales typical in ML. Furthermore, in structural bioinformatics such data generally relate to protein three-dimensional (3D) structures, which are inherently far more complex than sequence-based data. A significant and recurring challenge concerns the creation of large, high-quality, openly-accessible datasets that can be used for specific training and benchmarking tasks in ML pipelines for predictive modeling projects, along with reproducible splits for training and testing. Here, we report Prop3D, a protein biophysical and evolutionary featurization and data-processing pipeline that we have developed and deployed—both in the cloud and on local HPC resources—in order to systematically and reproducibly create comprehensive datasets, using the Highly Scalable Data Service (HSDS). Prop3D and its associated 'Prop3D-20sf' dataset can be of broader utility, as a community-wide resource, for other structure-related workflows, particularly for tasks that arise at the intersection of deep learning and classical structural bioinformatics.

Author Summary

We have developed a 'Prop3D' platform that allows for the creation, sharing and reuse of atomically-resolved physicochemical properties for any library of protein domains (preexisting or user-defined); in addition, we provide an associated 'Prop3D-20sf' protein dataset, obtained by applying this approach to CATH. This resource may be of use to the broader community in at least three ways. Firstly, the Prop3D workflow code can be customized and deployed on various computational platforms (cloud-based or local high-performance compute clusters), with scalability achieved largely by saving the results to distributed HDF5 files via the Highly Scalable Data Service (HSDS). Secondly, the linked Prop3D-20sf dataset provides a hand-crafted, already-featurized dataset of protein structures for 20 highly-populated CATH families; importantly, provision of this pre-computed resource can aid the more efficient development (and reproducible deployment) of ML pipelines. Thirdly, Prop3D-20sf's construction explicitly takes into account (in creating datasets and data-splits) the enigma of 'data leakage', stemming from the evolutionary relationships between proteins. The datasets that we provide (using HSDS) can be freely accessed via a standard representational state transfer (REST) application programming interface (API), along with accompanying Python wrappers for NumPy and the popular ML framework PyTorch.

1 Introduction

The advent of AlphaFold2 [1] and related deep learning approaches now enables us to access the 3D structure of virtually any protein sequence. As was the case for sequence-level data in the 1980s-2000s (enabled by technologies such as PCR and, ultimately, genome sequencing), 3D structural data has now been transformed into a readily available commodity. How might such a wealth of structural data inform our understanding of biology's central *sequence* ↔ *structure* ↔ *function* paradigm? Two new, post-AlphaFold2 challenges can be identified: (i) elucidating the *relationships* between all structures in the protein universe, and (ii) armed with millions of new protein structures [2], exploring the limits of protein *function prediction*. Arguably,

classic structural bioinformatics approaches, which are largely founded on comparative structural analyses, should now be an even more powerful tool in analyzing and accurately predicting protein function.

In structural bioinformatics, the 'data' center around biomolecular 3D structures. In this work, we take such 'data' to mean the geometric structures themselves, augmented by a multitude of other properties. These other properties can be (i) at potentially varying length-scales (atomic, residue-level, domains, etc.), and (ii) of numerous types, either *biological* in origin (e.g., phylogenetic conservation at a site) or *physicochemical* in nature (e.g., hydrophobicity or partial charge of an atom, concavity of a patch of surface residues, etc.). A significant and persistent challenge in developing and deploying ML workflows in structural bioinformatics concerns the availability of large, high-quality, openly-accessible datasets that can be (easily) used in large-scale analysis and predictive modeling projects. Here, 'high-quality' implies that specific training and benchmarking tasks can be performed reproducibly and without undue effort, and that the data splits for model training/testing/validation are reproducible. (A stronger requirement is that the split method also be at least semi-plausible, or not nonsensical, in terms of the underlying biology of a system—e.g., taking into account evolutionary relationships that muddle the assumed [statistical] independence of the splits; this topic of evolutionary 'data leakage' is discussed in more detail below.)

A common task in classical bioinformatics involves transferring functional annotations from a well-characterized protein to a protein of interest, if given sufficient shared evolutionary history between the two proteins. A conventional approach to this task typically applies sequence or structure comparison (e.g., via BLAST [3] or TM-Align [4], respectively) of a protein of interest to a database of all known proteins, followed by a somewhat manual process of 'copying' or grafting the previously annotated function into a new database record for the protein of interest. However, in the era of ML we can now try to go automatically from sequence or structure directly to functional annotations; the ML models can 'learn' these evolutionary relationships between proteins as part of the model, thereby obviating the more manual/tedious (and subjective) alignment-related steps.

However, ML workflows for working with proteins—and, in particular, protein 3D structures (and structure-associated data)—are far more challenging, from a technological and data-engineering perspective, than are many of the standard and more routine ML workflows designed to work with inputs in other domains (a text corpus in natural language processing, image data for classification tasks, etc.). Protein structures are more difficult to work with, from both a basic and applied ML perspective, for several types of reasons, including: (i) fundamentally, all proteins are related through evolution, thereby causing 'data leakage' [5]; (ii) raw/unprocessed protein structures are not always biophysically and chemically well-formed (e.g., atoms or entire residues may be missing) [6, 7]; (iii) somewhat related, some protein structures may 'stress-test' the resiliency of existing data structures by having, for instance, multiple rotamers/conformers at some sites; (iv) a protein's biophysical properties, which are not always included and learned in existing ML models, are just as critical (if not more so) as the raw 3D geometry itself; and (v) there are many different possible representational approaches/models of protein structures (volumetric data, contact graphs, etc.) that can yield different results. In short, protein data must be carefully inspected and processed before they can be successfully used and split in precise ways to create robust ML models.

Motivated by these challenges, this paper presents Prop3D, a new platform for processing and otherwise working with protein domain structures, building datasets with cleaned/prepared structures, pre-calculated biophysical and evolutionary properties, and different protein representations, along with ML-ready train/test splits.

We also supply the methods used to recreate the datasets provided with Prop3D; these calculations can be done in a distributed manner and read into the Prop3D framework for use in ML models. We describe this new software and its associated dataset (Prop3D-20sf), after first delineating some of the specific considerations that motivated and shaped Prop3D's design.

2 Motivating Factors: Data Leakage, Biophysical Properties, & Protein Representations

2.1 Evolutionary Data Leakage

ML with proteins is uniquely challenging because all naturally occurring proteins are interrelated via the biological processes of molecular evolution [8]. Therefore, randomly chosen train/test splits are meaningless, as there are bound to be crossover relationships between proteins (even if only distantly homologous), ultimately leading to overfitting of the ML model. Moreover, the available datasets are biased—they sample the protein universe in a highly non-uniform (or, rather, *non-representative*) manner (Figure 1)—which leads to biased ML models. For example, there are simply more 3D structures available in the Protein Data Bank (PDB [9]) for certain protein superfamilies because, for instance, some of those families were of specific (historical) interest to specific laboratories, certain types of proteins are more intrinsically amenable to crystallization (e.g., lysozyme), some might have been disproportionately more studied and structurally characterized because they are drug targets (e.g., kinases), certain protein families were preferentially selected for during evolution [10], and so on.

An approach to handle this type of inherent bias would be to create training and validation splits that ensure that no pairs of proteins with $\geq 20\%$ sequence identity occur on the same side of the split [11].

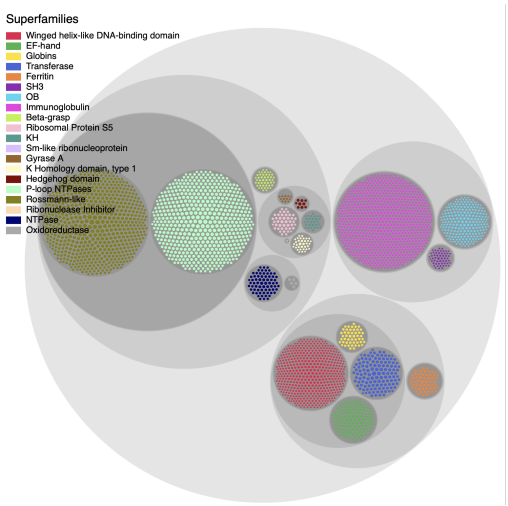


Fig 1. Uneven distribution of protein superfamiles. We show 20 superfamilies of interest to show how the number of known domains structures varies between superfamilies. The CATH hierarchy is shown as a circle packing chart.

In training ML models at the level of full (intact) protein chains, another source of bias in constructing training and validation sets stems from the phenomenon of domain re-use. This is an issue because many full protein chains are multi-domain

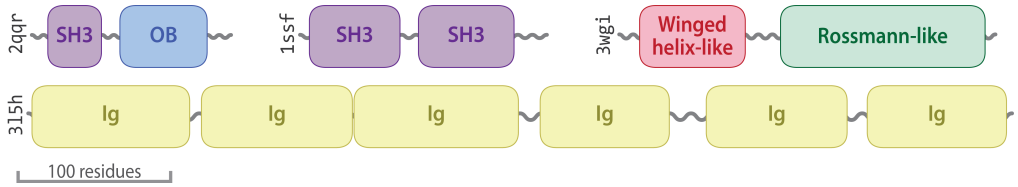


Fig 2. Data leakage and multi-domain proteins: A prime example of evolution-induced data leakage stems from the modular anatomy of many proteins, wherein multiple copies (which often only slightly vary—i.e., paralogous) of a particular domain are stitched together as part of the overall protein. This phenomenon is particularly prevalent among protein homologs from more phylogenetically recent species (e.g., eukaryotes like human or yeast, versus archaea or bacterial lineages). For example, many proteins that contain SH3, OB and Ig domains are found to include multiple copies of those domains.

(particularly true for proteins $\gtrapprox$ 120-150 residues), and many of those individual domains can share similar 3D structures (and functions) and be grouped, themselves, into distinct superfamilies. Some multi-domain proteins contain multiples of a given protein domain and so on; in other words, full-length proteins generally evolved so as to utilize individual domains in a highly modular manner (Figure 2). While splitting based on 20% sequence identity does limit this problem to some extent (if two domains have less than that level of similarity but are still from the same superfamily), a simple, straight-ahead split at a 20% (or whatever threshold) might negatively impact an ML algorithm at the very fundamental level of model training. In principle, note that this problem could also hold at the finer scale of shared structural fragments (i.e., sub-domain–level) too, giving rise to an even more difficult problem.

Possible approaches to mitigate these types of subtle biases would be to (i) create 'one-class' superfamily-specific models; or (ii) create multi-superfamily models, making sure to (a) over-sample proteins from under-represented classes and (b) under-sample proteins from over-represented classes [5, 12].

2.2 Biophysical Properties in ML

In many ML problems on proteins, it is useful to include biophysical properties mapped onto 3D locations of atoms and residues, thus providing a learning algorithm with additional sources of information. However, such properties are often ignored, as in sequence-based methods, which neglect 3D structure entirely and often only use a one-hot encoding of the sequence, perhaps augmented with some evolutionary information. In other cases, 3D structures are used and only the pure geometry of the atomic structure is used as input, neglecting the crucial biophysical properties that help define a protein's biochemical properties and physiological functions. There is also a trend in ML wherein one lets a model create its own embeddings, using only a small amount of hand-curated data (e.g., only atom type). Such approaches are generally taken because (i) it is expensive to calculate a full suite of biophysical properties for every atom, say on the scale of the entire PDB (≈200K structures); and (ii) the available models, theories and computational formalisms used to describe the biophysical properties of proteins (e.g., approximate electrostatics models, such as the generalized Born) may be insufficiently accurate, thereby adversely influencing the resultant ML models.

Irrespective of the specific details of one use-case or set of tasks versus another, we have found it useful to have available a database of pre-calculated biophysical properties. Among other benefits, such a database would enable one to: (i) save time

during the ML training process, by avoiding repetition of calculations that many others in the community may have already performed on exactly the same proteins (note that this also speaks to the key issue of reproducibility of an ML workflow or bioinformatics pipeline); and (ii) compare the predicted embeddings of the ML model to known biophysical properties, thereby providing a way to assess the accuracy and veracity of the ML model under development.

Some existing protein feature databases offer various biophysical properties of proteins at different structural 'levels' (atomic, residue-based, etc.), as shown in Table 1.

Dataset	Wikidata Entry	Domain Level	Residue Level	Atom Level	Residue-Residue Graph	2° Structure	Electrostatics & Charge	Surface & Curvature	Protein Interaction Sites	Train/Validation Splits	Clusters	Evolutionary Info	File format
PDB [9]	Q766195		✓	✓†		✓		✓			✓		Web, PDB & MMTF
UniProt [13]	Q905695		✓			✓					✓		Web & REST
CATH [14]	Q5008897	✓	✓	✓†							✓	✓	PDB & REST
FEATURE [15]	Q114878648		✓	✓		✓	✓	✓					ASCII
PredictProtein [16]	Q7239681		✓			✓		✓	✓			✓	Web, REST & JSON
DescribePROT [17]	Q111288739		✓					✓	✓			✓	Web & JSON
ATOM3D/DIPS [18]	Q114878673		✓	✓†					✓	✓	✓		JSON & PyTorch
ProteinNet [19]	Q114878717		✓	✓†		✓				✓	✓	✓	TensorFlow
SidechainNet [20]	Q114878822		✓	✓†		✓				✓	✓	✓	PyTorch & Pickle
Prop3D [this work]	**Q108040542**	✔	✔	✔	✔	✔	✔	✔		✔	✔	✔	**HDF, HSDS & PyTorch**

Table 1. A catalog of protein feature datasets that can be used in ML. Many different datasets of sequences, structures, and biophysical properties exist. They all contain different amounts of data, data on different levels/scales (chain, domain, residue, atom), and some contain biophysical properties attached to each atom and/or residue. The † denotes a database that uses atomic coordinates, but with no biophysical properties associated with those coordinates.

2.3 Protein Representations

There are various ways to computationally represent a protein for use in ML, each with its own strengths and weaknesses. Many protein structure & feature databases are 'hard-wired' so as to include data that can populate only one type of representation; however, to be flexible and agile (and therefore more usable), new databases and database-construction approaches need to allow facile methods to switch between various alternate representations of proteins (i.e., extensibility of structural representations). The remainder of this section describes approaches that have been used, wherein a protein is represented as a simple sequence, as a graph-based model (residue-residue contact networks), or as a 3D volumetric dataset. We now briefly consider each of these in turn.

2.3.1 Proteins in 1D: Sequences

The simplest approach to represent a protein is to treat it as a sequence of amino acids, ignoring all structural information. In ML workflows, the sequence is generally 'one-hot encoded', meaning that each individual character(/residue) in the sequence is

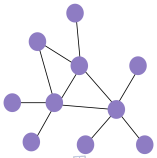
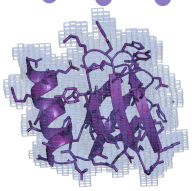
Dimensionality	Representation	Example
1D	Protein Sequence	MIANE...
2D	Residue-Residue Graph	
3D	Protein Structure 3D Volume	

Table 2. Some fundamentally different types of protein structure representations (reps) are schematized here, arranged by dimensionality of the rep. One can always traverse from higher- to lower-dimensional reps without loss of information, while the reverse is not true. Note that some types of reps are more amenable to encapsulation in simple data structures, e.g. protein sequences as character strings (built-in types for programming languages), and residue-residue graphs as adjacency matrices; that 3D structures are generally not as 'cleanly' representable (in 3D), via available data structures for use in ML workflows, motivates much of Prop3D's functionality.

attributed with a 20-element vector; in that vector, all elements are set to zero except for the index of the amino acid type that matches the current position, which is set to one. Biophysical properties can also be appended to this representation, giving a feature vector.

2.3.2 Proteins in 2D: Residue-residue Graphs

A conceptually straightforward method to capture a protein 3D structure is to build a graph, using the amino acid residues as vertices and inter-atomic contacts between those residues (near in 3D space) as edges. Individual nodes can be attributed with the one-hot encoding of the residue type along with biophysical properties, and to each edge can be attributed geometric properties such as a simple Euclidean distance (e.g., between the two residues/nodes), an angle of interest (defined by three atoms), any dihedral angles that one likes (defined by four atoms), and so on. These graphs can be fully connected, e.g. all residues are connected to one another, or they may include edges only between residues that lie within a certain cutoff distance of one another (e.g. 5 Å).

2.3.3 Proteins in 3D: Structures as 3D Volumetric Data

Another approach to handle protein structures in ML is to treat them as spatially discretized 3D images, wherein volumetric elements ('voxels') that intersect with an atom are attributed with biophysical properties of the overlapping atom. Here, note that one must define '*an atom*' precisely, e.g. as a sphere of a given van der Waals radius, centered at a specific point in space (the atom's coordinates), such that the notion of "*intersection* with a specific voxel" is well-defined. Early work in deep neural nets used these types of structural representations, though the volumetric approaches have been less prevalent recently for reasons that include: (i) size constraints (large proteins consume much memory, scaling with the cube of protein size, in terms of number of residues); (ii) mathematical considerations, such as this representation's lack of rotational invariance (e.g., structures are manually rotated); (iii) fixed-grid

volumetric models are inherently less flexible than graph representations (e.g. 3D images are static and cannot easily incorporate fluctuations, imparting a 'brittleness' to these types of data structures); and (iv) related to the issue of brittleness, there exists a rich and versatile family of graph-based algorithms, versus more limited (and less easily implemented) approaches for discretized, volumetric data. However, 3D volumetric approaches have at least two benefits: (i) 3D representations offer a more natural way for humans to visually analyze a protein and 'hold' the object in mind, versus even 2D graph-based approaches. (ii) The form taken by the data in a 3D volumetric representation is more amenable to 'explainable' AI/ML algorithmic approaches, such as layer-wise relevance propagation, whereby any voxels identified by the algorithm as being 'important' can be readily mapped back to specific atoms, residues, patches, etc. (which may, in turn, be of biochemical or functional interest); such operations are not as readily formulated with 1D (sequence) or 2D (graph) representations.

A common approach to voxelize a protein structure into a dense grid is to calculate the distance of every atom to every voxel, then use the Lennard-Jones potential to map scaled biophysical properties to each voxel [21, 22]. This works well for small proteins, but can take a long time for larger structures because of the $\mathcal{O}(n^2)$ run-time. A faster voxelization approach is to create a sparse grid, preserving only voxels that overlap with a van der Waals envelope around each atom. This can be done using kD-trees, with a resultant advantage of scaling as $\mathcal{O}(n \log n)$ [12].

When treating proteins as 3D images for purposes of training ML models, one must take into account the importance of rotational invariance. All protein structures must be rotated to achieve (ideally) uniform random sampling, which can be achieved via the 3D rotation group ($SO(3)$), formulated as a Haar distribution over unit quaternions [23]; however, those numerical steps add significant computational overhead unless the model is already rotationally invariant, such as with equivariant neural networks [24].

2.4 Outline of This Work

The remainder of this work presents Prop3D, a new protein domain structure dataset that includes (i) corrected/sanitized protein 3D structures, (ii) biophysical properties for each atom and residue, to allow for multiple representation modes, as well as (iii) pre-constructed train, test & validation splits that have been specifically formulated for use in ML of proteins (to mitigate evolutionary data leakage).

3 Overview of the Software & Dataset

3.1 Architecture and Design

The Prop3D-20sf dataset is created by using Prop3D in tandem with two other frameworks that we developed: (i) 'Meadowlark', for processing and interrogating individual protein structures and (ii) 'AtomicToil', for creation of massively parallel workflows of many thousands of structures. An overview of these tools is given in Figure 3. While each of these codebases are intricately woven together (in practice), giving the Prop3D functionality, it helps to consider them separately when examining their utility/capabilities and their respective roles in an overall Prop3D-based ML pipeline.

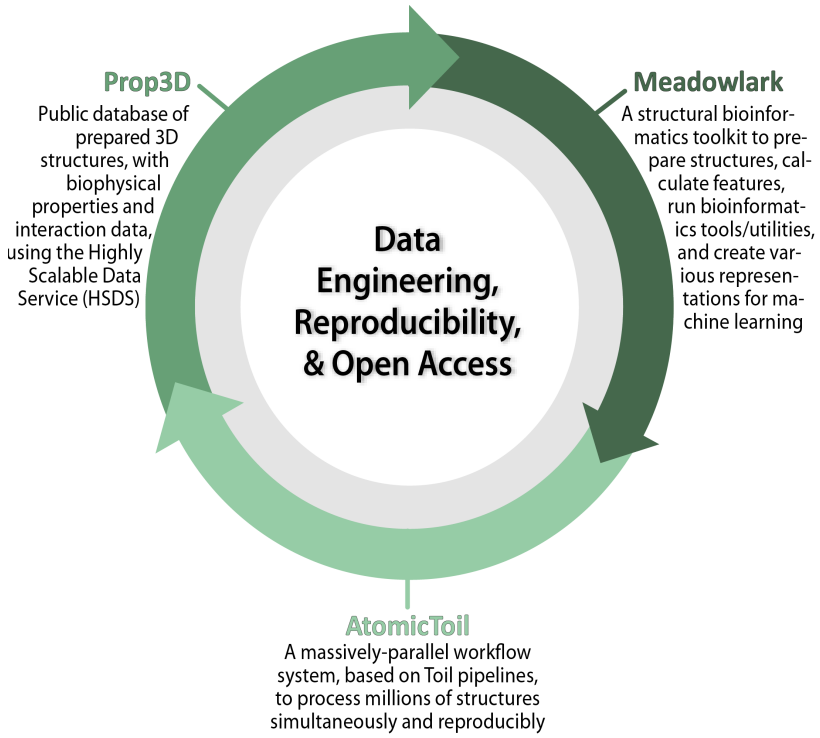


Fig 3. Overview of Prop3D Components. We developed Prop3D as a framework to create and share biophysical properties. This was accomplished by creating two separate packages within the framework: (i) 'Meadowlark', to prepare structures, calculate features, and run Docker-ized tools; and (ii) 'AtomicToil', to run calculations in parallel and on the cloud using Toil. The datasets we built via Prop3D are available as a publicly-available HSDS endpoint.

3.2 Meadowlark: An Extensible, Dockerized Toolkit for Reproducible, Cross-platform Structural Bioinformatics Workflows

In bioinformatics and computational biology more broadly, many tools and codes can be less than straightforward to install and operate locally: they each require various combinations of operating system configurations, specific versions of different languages and libraries (which may or may not be cross-compatible), have various interdependencies for installation/compilation (and for run-time execution), and so on. Moreover, considered across the community as a whole, researchers spend many hours installing (and perhaps even performance-tuning) these tools themselves, only to find that they are conducting similar development and upkeep of this computational infrastructure as are numerous other individuals; all the while, the data, results, and technical/methodological details underpinning the execution of a computational pipeline are typically never shared, at least not before the point of eventual publication (i.e., months to even years after the point at which it would have been most useful to others). Following the examples of the UC Santa Cruz Computational Genomics Laboratory (UCSC-CGL) and Global Alliance for Genomics & Health (GA4GH) [25], we Docker-ize common structural bioinformatics tools to make them easily deployable and executable on any machine, along with parsers to handle their outputs, all without leaving a top-level Python-based workflow. New software can be

added into meadowlark if it exists as a Docker or Singularity container [26,27]. For a list of codes and software tools that we have thus far made available, see Tables 3 & 4 or visit our Docker Hub for the most current information.

3.3 AtomicToil: Mapping Structural Information to Sets of Massively Parallel Tasks

To construct and automate the deployment of massively parallel workflows in the cloud, we use a Python-based workflow management system (WMS) known as Toil [26]. Each top-level Toil job has child jobs and follow-on jobs, enabling the construction of complex MapReduce-like pipelines. A Toil workflow can be controlled locally, on the cloud (e.g., AWS, Kubernetes), or on a compute farm or a high-performance computing platform such as a Linux-based cluster (with a scheduler such as SLURM, Oracle Grid Engine, or the like). Background information on the data-flow paradigm, flow-based programming, task-oriented toolkits (like Toil) and related WMS concepts, as they pertain to bioinformatics, can be found in [28].

Within Prop3D, we specifically created multiple ways to instantiate a workflow:

1. Based on PDB files (which can contain multiple chains or a single domain), to systemically map PDB files to jobs in order to run a given function (or 'apply' the function, in the parlance of functional programming).

2. Based on CATH's hierarchical organization, where one job is created for each entry in the CATH hierarchy, with child jobs spawned for subsidiary levels in the hierarchy. Once the workflow reaches a job for each individual domain (or specified level), it will run a given, user-provisioned function.

New functions can be added into a workflow by defining new Toil job functions, which can be as simple as standalone Python functions with specific, well-formed signatures (call/return semantics).

3.4 Capabilities and Features

In this section we provide two examples of Prop3D usage, one relatively simple and the other more advanced: (i) protein structure preparation, and (ii) biophysical property calculations (and annotation). While not included here, we note that Prop3D also may be useful in creating workflows to build and validate intermolecular associations, e.g. to study domain-domain interactions and protein complexes.

3.4.1 Protein Structure Preparation

We 'clean' or 'sanitize' a starting protein structure by applying the following scheme: We begin by selecting the first model (from among multiple possible models in a PDB file), the correct chain, and the first alternate location (if multiple conformers are present for an atom/residue). Next, we remove hetero-atoms (water or buffer molecules, other crystallization reagents, etc.); these steps are achieved via pdb-tools [29]. Finally, we then modify each domain structure via the following stages: (i) Build/model any missing residues with MODELLER [30]; (ii) Correct/optimize rotamers with SCWRL4 [31], e.g. if there are any missing atoms; and (3) Add hydrogens and perform a rough potential energy minimization with the PDB2PQR toolkit [32]. Note that this general workflow, schematized in Figure 4, was applied in constructing the Prop3D-20sf dataset.

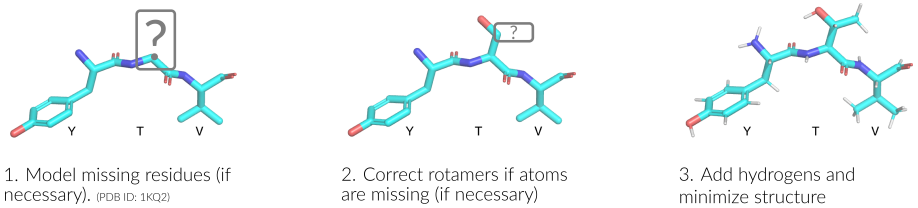


Fig 4. Protein Preparation. In working with protein structures, e.g. in constructing the Prop3D-20sf dataset, every domain is 'corrected' by adding missing atoms and residues, checking rotameric states, protonating, and performing a crude potential energy minimization of the 3D structure.

3.4.2 Biophysical Property Calculation & Featurization

The Prop3D toolkit enables one to rapidly and efficiently compute biophysical properties for all structural entities (atoms, residues, etc.) in a dataset of 3D structures (e.g., from the PDB or CATH), and then map those values onto the respective entities as features for ML model training or other downstream analyses.

For atom-level features, we create one-hot encodings for 23 AutoDock atom names, 16 element names, and 21 residue types (20 standard amino acids and one UNKnown placeholder). We also include van der Waals radii, charges from PDB2PQR [32], electrostatic potentials computed via APBS [33], concavity values that we calculate via CX [34], various hydrophobicity features of the residue that an atom belongs to (Kyte-Doolite [35], Biological [36] and Octanol [37]), and two measures of accessible surface area (per-atom, via FreeSASA [38], and per-residue via DSSP [39]). We also include different types of secondary structure information: one-hot encodings for DSSP [39] 3- and 7- secondary structure classifications, as well as the backbone torsion angles (ϕ, ψ; along with embedded sine and cosine transformations of each). We also annotate aromaticity, and hydrogen bond acceptors and donors, based on AutoDock atom-name types. As a gauge of phylogenetic conservation, we include sequence entropy scores from EPPIC [40]. These biophysical, physicochemical, structural, and phylogenetic features are summarized in Figure 5 and are exhaustively enumerated in Table 6. Finally, we also provide functionality to create discretized values of features via the application of Boolean logic operators to the corresponding continuous-valued quantities of a given descriptor, using simple numerical thresholding (Table 7).

Some of the above properties are computed at the residue level and mapped to each atom in the residue (e.g., hydrophobicity is one such property); that is, the 'child' atom inherits the value of the given feature from its 'parent' residue. For other features, residue-level values are calculated by combining atomic quantities, via various summation or averaging operations applied to the properties' numerical values (as detailed in Table 6).

3.5 Dataset Design and Open Data Format

To handle the large amount of protein data in our massively parallel workflows, we employ the Hierarchical Data Format (HDF5 [41]), along with the Highly Scalable Data Service (HSDS). The HDF5 file format is a useful way to store and access large protein datasets because it allows us to chunk and compresses the CATH protein structure hierarchy in a scalable and efficient manner. Using this approach versus, for example, creating many individual files spread across multiple directories, we combine them into a 'single' file/object that is easily shareable and can be accessed via a hierarchical structure of groups and datasets, each with attached metadata descriptors.

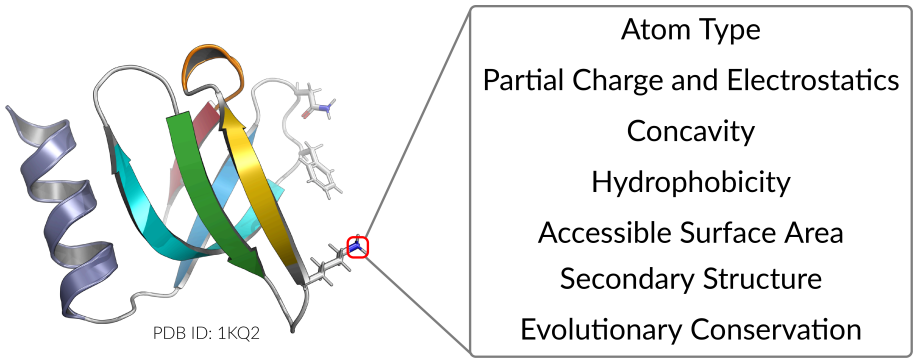


Fig 5. Calculated Biophysical Properties. For each protein domain, we annotate every atom with the following features: atom type, element type, residue type, partial charge & electrostatics, concavity, hydrophobicity, accessible surface area, secondary structure type, and evolutionary conservation. For a full list of features, see the text and Tables 6 & 7.

Moreover, the HSDS extension to this file-format system allows multiple readers and writers which, in combination with Toil, affords a degree of parallelization that makes it extremely fast to create new datasets.

We note that many computational biologists are migrating to HDF5 [42–44] and HSDS [45] because binary data can be rapidly read, are easily shareable, and this system provides well-integrated metadata and other beneficial services, schema and features (thus, e.g., facilitating attribution of data provenance). Before HDF5(/HSDS) and other binary formats came into use, biological data formats for protein structure relied on human-readable ASCII files. Legacy PDB files have been the *de facto* format to store protein structures. Originally developed in 1976 to work with punch cards, legacy PDB files are ASCII files that have a fixed-column width and maximally 80 characters per line [46]. A structure could be attributed with only one type of biophysical property, by substituting the values in the B-factor column—a highly limited workaround. Due to the inflexibility of the legacy PDB file format and its limitations as a data exchange format, the macromolecular Crystallographic Information Framework (mmCIF) was developed; this file format was more extensibly structured and allowed for a plethora of biophysical properties [47]. Most recently, spurred by the slow nature of reading ASCII files, the Macromolecular Transmission Format (MMTF) has been developed to store protein structures in a compact binary format, based on MessagePack format (version 5) [48, 49]. While the MMTF is almost ideal for use in ML tasks, it still relies on individual files in a file system with no efficient, *distributed* mechanism to read in all files, no way to include metadata higher than residue level, and no ability to combine train/test splits directly into the schema—which were some of our motivating factors in adopting HDF5 and HSDS.

For Prop3D and Prop3D-20sf, the HDF5 file is built by starting with the CATH database, which provides a hierarchical schema (namely, *Class* $\supset$ *Architecture* $\supset$ *Topology* $\supset$ *Homologous Superfamily*) that is naturally amenable to parallelization and efficient data traversal, as shown in Figure 6. In Prop3D, a superfamily can be accessed by its CATH code as the group key (e.g., '2/60/40/10' for Immunoglobulin). We then split each superfamily into two groups (Figure 6): (i) '`domains`', containing groups for each domain inside that superfamily, and (ii) '`data_splits`', containing pre-computed train (80%), validation (10%), and test (10%) data splits for use in ML models, where each domain in each split is hard-linked to the group for that domain (dashed green arrows in Figure 6). Each domain group contains datasets for different types of

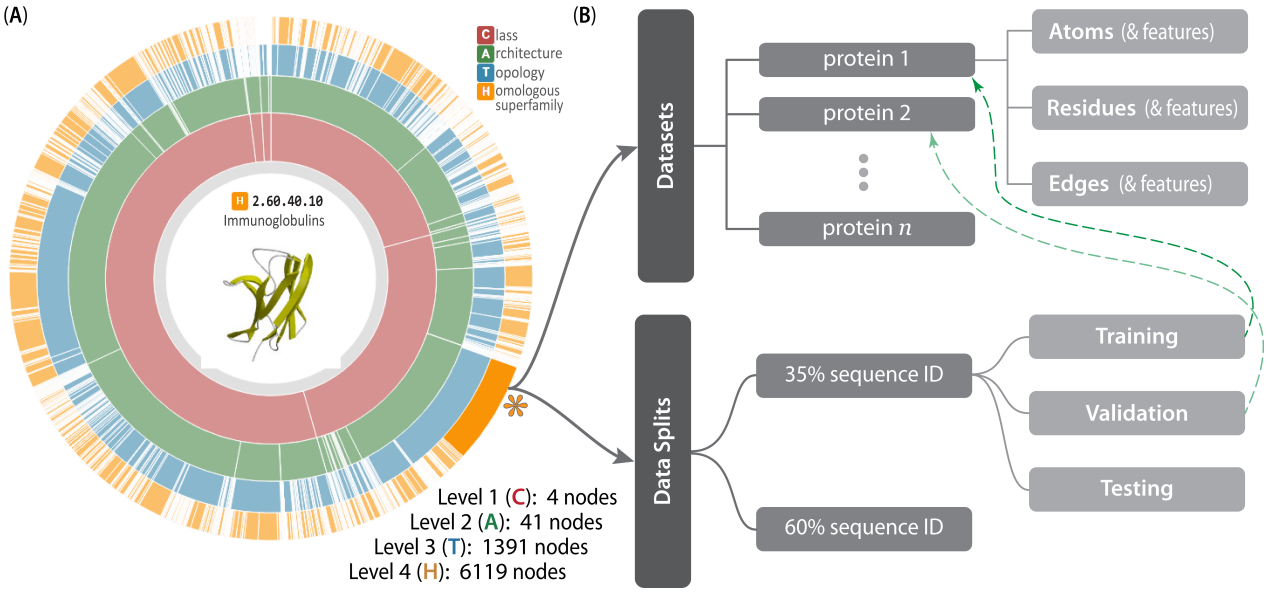


Fig 6. The Hierarchical Structure of Prop3D. The inherently hierarchical structure of CATH (A) is echoed in the design schema underlying the Prop3D dataset (B), as illustrated here. Prop3D can be accessed as an HDF5 file seeded with the CATH hierarchy for all available superfamilies. For clarity, an example of one such superfamily is the individual H-group `2.60.40.10` (Immunoglobulins) shown here as the orange sector (denoted by an asterisk near 4 o'clock). Each such superfamily is further split into (i) the domain groups, with datasets provided for each domain (atomic features, residue features, and edge features), as delineated in the upper-half of (B), and (ii) pre-calculated data splits, shown in the lower-half of (B), which exist as hard-links (denoted as dashed green lines) to domain groups. (The 'sunburst' style CATH diagram, from cathdb.info, is under the Creative Commons Attribution 4.0 International License.)

features: 'Atoms', 'Residues' and 'Edges'. The 'Atoms' dataset contains information from the PDB ATOM field as well all of the biophysical properties calculated for each atom. 'Residues' contains biophysical properties of each residue and position (average of all of its daughter atoms), e.g. for use in coarse-grained models. Finally, 'Edges' contains properties for each *residue* ↔ *residue* interaction, thereby enabling the construction and annotation of contact maps in graph-based representations/models.

In terms of the data-processing pipeline, HSDS allows HDF5 data stores to be hosted in S3-like buckets, such as AWS or MinIO, remotely and with accessibility achieved via a REST API. HSDS data nodes and service nodes (Figure 7) are controlled via a load-balancer in Kubernetes in order to enable efficient, distributed mechanisms to query HDF5 data stores, as well as write data with a quick, efficient, distributed mechanism; this is achieved via various means, including using data-caching and implicit parallelization of the task mapping (see Fig. 7). HSDS allows for multiple readers and multiple writers to read or write to the same file simultaneously, using a 'distributed' HDF5 multi-reader/multi-writer Python library known as h5pyd (Figure 7). We have set-up a local k3s instance, which is an easy-to-install, lightweight distribution of Kubernetes that can run on a single machine along with MinIO S3 buckets. This approach is particularly useful for purposes of enabling flexible scalability: our solution works on data infrastructures that can be both big and (relatively) small.

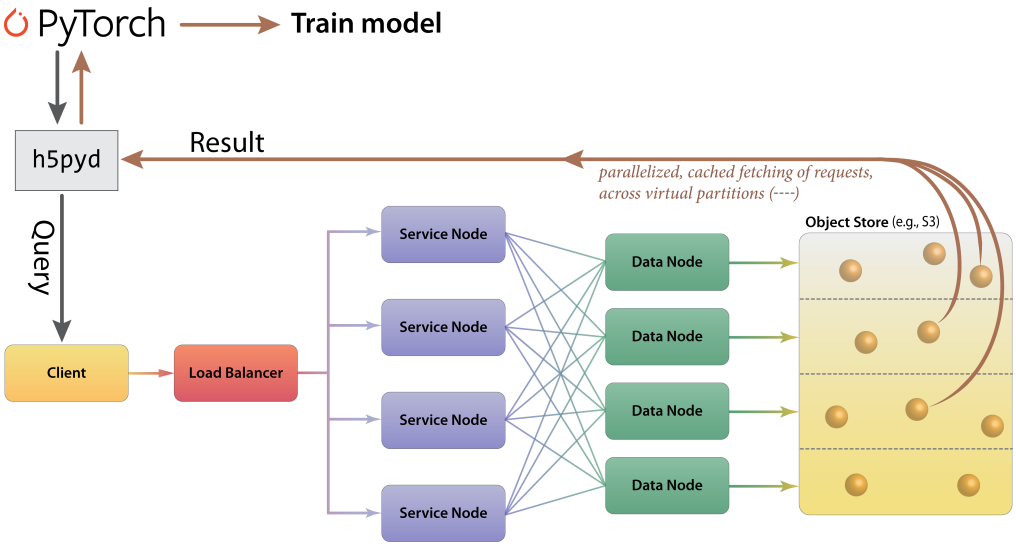


Fig 7. Cloud-based Access to The Prop3D Dataset with HSDS. HSDS creates Service Nodes, which are containers that handle query requests from clients, and Data Nodes, which are containers that access the object storage in an efficient distributed manner. The Prop3D dataset can be used as input to train an ML model by either accessing the data via the Python library h5pyd or through the DeepUrfold Python package that contains PyTorch data loaders [12]. This illustration was adapted from one which can be found at the HSDS webpage (available under an Apache 2.0 license, which is compatible with CC-by-4.0).

In creating the Prop3D-20sf dataset, HSDS, in combination with a Toil-enabled workflow, allows for each parallel task to write to the same HDF5 data store simultaneously. The Prop3D-20sf dataset can be read in parallel as well, e.g. in PyTorch. We provide PyTorch Data Loaders to read the Prop3D dataset from an HSDS endpoint using multiple processes, available in our related DeepUrfold Python package [12]. When HSDS was used as a system for distributed training of deep learning models (in our DeepUrfold workflow), as opposed to using raw ASCII files, a speedup of 8 hours was achieved (from 24 hours to 16 hours of wall-clock time) for training an immunoglobulin-specific variational autoencoder model with 25,524 featurized immunoglobulin domain structures (Figure 8). Thus, we found it clearly and significantly advantageous to utilize the parallelizable data-handler capacity provided by a remote, cloud-based, parallel-processing system like HSDS.

4 Data Availability & Prop3D's FAIRness

In general, the creation of scalable, reproducible scientific workflows faces challenges that stem from the sheer volume and heterogeneity of available data-sources and data-types, in addition to potential other factors, such as the variable range of computing platforms, architectures and capabilities that one may seek to deploy a workflow across (e.g., multi-core processing on a local workstation, versus a Linux HPC cluster, versus an eScience grid or other highly distributed network environment in the cloud). The 'FAIR' principles for scientific data provide a set of best-practices that contribute to the research enterprise by striving to make datasets *Findable*, *Accessible*, *Interoperable*, and *Reproducible* [50]. In other words, FAIR datasets should be (i) easy to *find*, with appropriate metadata to facilitate searching by others; (ii) one

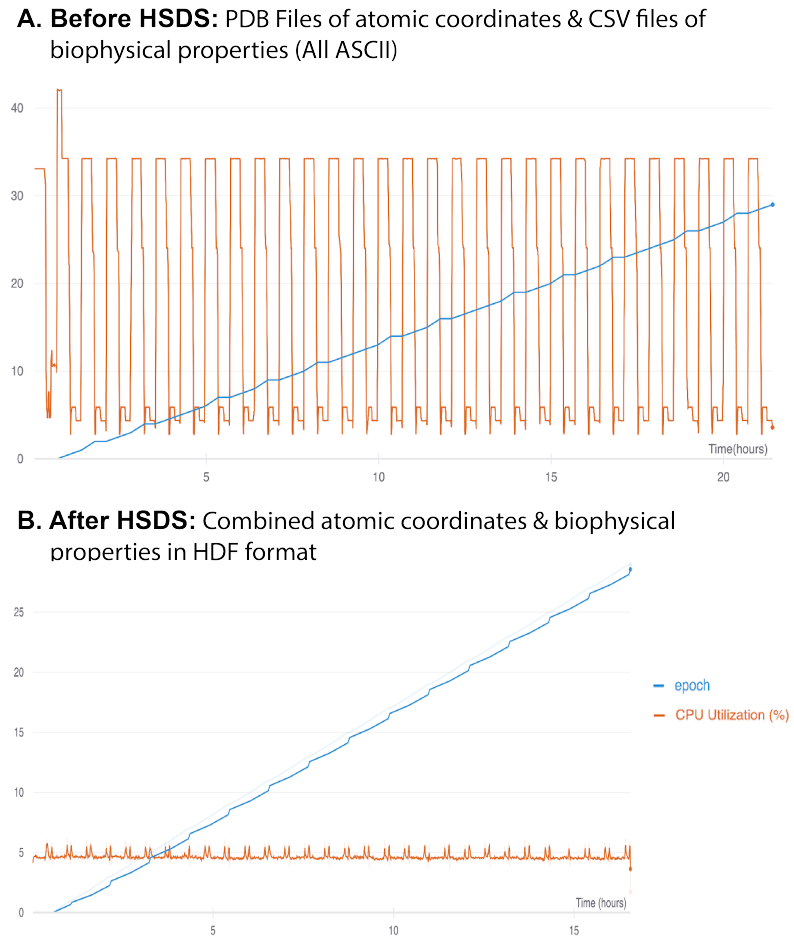


Fig 8. HSDS Affords Improved Training Runtimes. We trained an immunoglobulin-specific variational autoencoder with ≈25K domain structures, using 64 CPUs to process data and four GPUs for 30 epochs [12]. **A.** Before we implemented HSDS, we stored domain structures as simple, text-based PDB files (parsed with BioPython) along with the corresponding biophysical properties for all atoms in these PDB structures as plaintext CSV files (parsed with Pandas). That process took ≈24 hours of wallclock time for ≈50K ASCII files. **B.** Reformulating and streamlining the pipeline with HSDS yielded an ≈30% speed-up: training runtimes improved by ≈8 hours (to an ≈16 hours total), with there being more efficient CPU usage while reading all of the data (orange traces). These data-panel images were exported from our Weights and Biases training dashboard.

should be able to *access* all of the data easily; (iii) one should be able to integrate and otherwise *interoperate* the data with other data-sources and software frameworks; and (iv) the data should be (re)usable and *replicable* by others (a bedrock of the scientific method). When possible, these guidelines would apply equally well to both datasets themselves as well as to the code that underlies the data-processing and data-generating pipelines (i.e., the software framework would be FAIR-compliant, insofar as its resultant data are FAIR).

As summarized in the remainder of this section (and detailed in the Supplemental Material), we have sought to make Prop3D FAIR by providing unique identifiers and

searchable metadata for open platforms such as Zenodo, WikiData, the Open Science Foundation, and the University of Virginia School of Data Science's Open Data Portal.

The Prop3D code, to run predefined workflows, exists in our GitHub repository (https://github.com/bouralab/Prop3D) with scripts provided to set-up HSDS and Kubernetes, e.g. if running on one's own local system via k3s.

The pre-calculated features and data splits for 20 CATH superfamlies (i.e., Prop3D-20sf) exist in our HSDS endpoint at the University of Virginia (http://hsds.uvarc.io/bournelab/Prop3D.h5) with the raw HDF5 data also available on Zenodo (https://doi.org/10.5281/zenodo.6873024). These features can be read into a Python program using either h5pyd, our Prop3D library (https://github.com/bouralab/Prop3D), or through custom PyTorch Data Loaders that are available in our DeepUrfold GitHub repository (https://github.com/bouralab/DeepUrfold). Finally, all of our Docker-ized tools can also be obtained from our Docker Hub at https://hub.docker.com/u/edraizen.

We have used Wikidata to cite the software we use, as well as to create links to the code and data repositories reported herein (Q108040542).

5 Summary & Outlook

This work has presented Prop3D, a modular utility for large-scale protein properties featurization and associated data-processing. Developed and deployed as part of an ML pipeline in another project [12], we have engineered the Prop3D platform to be extensible and scalable. This tool can be used with local HPC resources as well as in the cloud, and allows one to systematically and reproducibly create comprehensive datasets using the Highly Scalable Data Service (HSDS). We have used Prop3D to create (and share) a new 'Prop3D-20sf' resource; this protein dataset, freely available as an HSDS endpoint, combines 3D coordinates with biophysical and evolutionary properties (for each atom), in each structural domain for 20 highly-populated homologous superfamilies in CATH. These 3D domains are sanitized via numerous steps, including clean-up of the covalent structure (e.g., adding missing atoms and residues) and physicochemical properties (protonation and energy minimization). Our database schema mirrors CATH's hierarchy, mapped to a system based on HDF5 files and including atomic-level features, residue-level features, residue-residue contacts, and pre-calculated train/test/validate splits (in ratios of 80/10/10) for each superfamily derived from CATH's sequence-identity–based clusters (e.g., 'S35' for groups of proteins culled at 35% sequence identity). We believe that Prop3D-20sf, and its underlying Prop3D framework, may be useful as a community resource in developing workflows that entail processing protein 3D structural information, particularly for pipelines that arise at the intersection of machine learning and structural bioinformatics. Our construction of Prop3D-20sf sought to directly and explicitly address the issue of evolutionary data leakage, thereby hopefully mitigating any bias in ML models trained with these datasets.

The Prop3D approach and its attendant Prop3D-20sf pre-computed dataset can be used to compare sequence-based (1D), residue-contact–based graphs (2D), and structure-based (3D) methods. For example, one could imagine training a supervised model, with input being a protein sequence, to predict a specific residue-based biophysical property. Similarly, unsupervised models can be trained using one or all of the biophysical properties to learn protein embeddings, such as was the case in our DeepUrfold project [12].

We built AtomicToil to enable the facile creation of reproducible workflows, starting with PDB files or by traversing the CATH hierarchy, as well as the Meadowlark toolkit to run Docker-ized structural bioinformatics software. While we primarily developed

these tools in order to create the Prop3D-20sf dataset, we envision that the toolkit can be integrated into feature-rich, standalone structural bioinformatics platforms, e.g. BioPython or Biotite.

Acknowledgements

We thank Lane Rasberry for critiquing the manuscript and providing support for Wikidata, and we appreciate the early efforts of Menuka Jaiswal, Saad Saleem and Yonghyeon Kweon on this project. Portions of this work were supported by the University of Virginia and NSF Career award MCB-1350957. EJD was supported by a University of Virginia Presidential Fellowship in Data Science.

References

1. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, et al. Highly accurate protein structure prediction with AlphaFold. Nature. 2021;596(7873):583–589. doi:10.1038/s41586-021-03819-2.

2. Varadi M, Anyango S, Deshpande M, Nair S, Natassia C, Yordanova G, et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. Nucleic Acids Research. 2021;50(D1):D439–D444. doi:10.1093/nar/gkab1061.

3. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research. 1997;25(17):3389–3402. doi:10.1093/nar/25.17.3389.

4. Zhang Y, Skolnick J. TM-align: a protein structure alignment algorithm based on the TM-score. Nucleic Acids Research. 2005;33(7):2302–2309. doi:10.1093/nar/gki524.

5. Whalen S, Schreiber J, Noble WS, Pollard KS. Navigating the pitfalls of applying machine learning in genomics. Nature Reviews Genetics. 2021;23(3):169–181. doi:10.1038/s41576-021-00434-9.

6. Joosten RP, Long F, Murshudov GN, Perrakis A. The PDB_REDO server for macromolecular structure model optimization. IUCrJ. 2014;1(4):213–220. doi:10.1107/s2052252514009324.

7. Eastman P, Swails J, Chodera JD, McGibbon RT, Zhao Y, Beauchamp KA, et al. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. PLOS Computational Biology. 2017;13(7):e1005659. doi:10.1371/journal.pcbi.1005659.

8. Graur D, Li WH. Fundamentals of molecular evolution. 2nd ed. New York, NY: Oxford University Press; 1999.

9. Burley SK, Bhikadiya C, Bi C, Bittrich S, Chen L, Crichlow GV, et al. RCSB Protein Data Bank: powerful new tools for exploring 3D structures of biological macromolecules for basic and applied research and education in fundamental biology, biomedicine, biotechnology, bioengineering and energy sciences. Nucleic Acids Research. 2020;49(D1):D437–D451. doi:10.1093/nar/gkaa1038.

10. Riesselman AJ, Ingraham JB, Marks DS. Deep generative models of genetic variation capture the effects of mutations. Nature Methods. 2018;15(10):816–822. doi:10.1038/s41592-018-0138-4.

11. Walsh I, Pollastri G, Tosatto SCE. Correct machine learning on protein sequences: a peer-reviewing perspective. Brief Bioinform. 2016;17(5):831–840.

12. Draizen EJ, Veretnik S, Mura C, Bourne PE. Deep Generative Models of Protein Structure Uncover Distant Relationships Across a Continuous Fold Space. bioRxiv. 2022;doi:10.1101/2022.07.29.501943.

13. The UniProt Consortium. UniProt: The Universal Protein Knowledgebase in 2021. Nucleic Acids Research. 2020;49(D1):D480–D489. doi:10.1093/nar/gkaa1100.

14. Sillitoe I, Bordin N, Dawson N, Waman VP, Ashford P, Scholes HM, et al. CATH: increased structural coverage of functional space. Nucleic Acids Research. 2020;49(D1):D266–D273. doi:10.1093/nar/gkaa1079.

15. Halperin I, Glazer DS, Wu S, Altman RB. The FEATURE framework for protein function annotation: modeling new functions, improving performance, and extending to novel applications. BMC Genomics. 2008;9(S2). doi:10.1186/1471-2164-9-s2-s2.

16. Bernhofer M, Dallago C, Karl T, Satagopam V, Heinzinger M, Littmann M, et al. PredictProtein - Predicting Protein Structure and Function for 29 Years. Nucleic Acids Research. 2021;49(W1):W535–W540. doi:10.1093/nar/gkab354.

17. Zhao B, Katuwawala A, Oldfield CJ, Dunker AK, Faraggi E, Gsponer J, et al. DescribePROT: database of amino acid-level protein structure and function predictions. Nucleic Acids Research. 2020;49(D1):D298–D308. doi:10.1093/nar/gkaa931.

18. Townshend RJL, Vögele M, Suriana P, Derry A, Powers A, Laloudakis Y, et al. ATOM3D: Tasks On Molecules in Three Dimensions. arXiv. 2020;doi:10.48550/ARXIV.2012.04035.

19. AlQuraishi M. ProteinNet: a standardized data set for machine learning of protein structure. BMC Bioinformatics. 2019;20(1). doi:10.1186/s12859-019-2932-0.

20. King JE, Koes DR. SidechainNet: An All-Atom Protein Structure Dataset for Machine Learning; 2020. Available from: https://arxiv.org/abs/2010.08162.

21. Jiménez J, Doerr S, Martínez-Rosell G, Rose AS, Fabritiis GD. DeepSite: protein-binding site predictor using 3D-convolutional neural networks. Bioinformatics. 2017;33(19):3036–3042. doi:10.1093/bioinformatics/btx350.

22. Simonovsky M, Meyers J. DeeplyTough: Learning Structural Comparison of Protein Binding Sites. Journal of Chemical Information and Modeling. 2020;60(4):2356–2366. doi:10.1021/acs.jcim.9b00554.

23. Rummler H. On the distribution of rotation angles: How great is the mean rotation angle of a random rotation? The Mathematical Intelligencer. 2002;24(4):6–11. doi:10.1007/bf03025318.

24. Fuchs FB, Worrall DE, Fischer V, Welling M. SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks. CoRR. 2020;abs/2006.10503.

25. Yuen D, Cabansay L, Duncan A, Luu G, Hogue G, Overbeck C, et al. The Dockstore: enhancing a community platform for sharing reproducible and accessible computational protocols. Nucleic Acids Research. 2021;49(W1):W624–W632. doi:10.1093/nar/gkab346.

26. Vivian J, Rao AA, Nothaft FA, Ketchum C, Armstrong J, Novak A, et al. Toil enables reproducible, open source, big biomedical data analyses. Nature Biotechnology. 2017;35(4):314–316. doi:10.1038/nbt.3772.

27. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. PLOS ONE. 2017;12(5):e0177459. doi:10.1371/journal.pone.0177459.

28. Cieślik M, Mura C. A lightweight, flow-based toolkit for parallel and distributed bioinformatics pipelines. BMC Bioinformatics. 2011;12:61.

29. Rodrigues J, Teixeira J, Trellet M, Bonvin A. pdb-tools: a swiss army knife for molecular structures. F1000Research. 2018;7(1961). doi:10.12688/f1000research.17456.1.

30. Webb B, Sali A. Comparative Protein Structure Modeling Using MODELLER. Current Protocols in Bioinformatics. 2016;54(1). doi:10.1002/cpbi.3.

31. Krivov GG, Shapovalov MV, Dunbrack RL. Improved prediction of protein side-chain conformations with SCWRL4. Proteins: Structure, Function, and Bioinformatics. 2009;77(4):778–795. doi:10.1002/prot.22488.

32. Dolinsky TJ, Czodrowski P, Li H, Nielsen JE, Jensen JH, Klebe G, et al. PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. Nucleic Acids Research. 2007;35(Web Server):W522–W525. doi:10.1093/nar/gkm276.

33. Jurrus E, Engel D, Star K, Monson K, Brandi J, Felberg LE, et al. Improvements to the scpAPBS/scp biomolecular solvation software suite. Protein Science. 2017;27(1):112–128. doi:10.1002/pro.3280.

34. Pintar A, Carugo O, Pongor S. CX, an algorithm that identifies protruding atoms in proteins. Bioinformatics. 2002;18(7):980–984. doi:10.1093/bioinformatics/18.7.980.

35. Kyte J, Doolittle RF. A simple method for displaying the hydropathic character of a protein. Journal of Molecular Biology. 1982;157(1):105–132. doi:10.1016/0022-2836(82)90515-0.

36. Hessa T, Kim H, Bihlmaier K, Lundin C, Boekel J, Andersson H, et al. Recognition of transmembrane helices by the endoplasmic reticulum translocon. Nature. 2005;433(7024):377–381. doi:10.1038/nature03216.

37. Wimley WC, White SH. Experimentally determined hydrophobicity scale for proteins at membrane interfaces. Nature Structural &amp Molecular Biology. 1996;3(10):842–848. doi:10.1038/nsb1096-842.

38. Mitternacht S. FreeSASA: An open source C library for solvent accessible surface area calculations. F1000Research. 2016;5:189. doi:10.12688/f1000research.7931.1.

39. Kabsch W, Sander C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983;22(12):2577–2637. doi:10.1002/bip.360221211.

40. Bliven S, Lafita A, Parker A, Capitani G, Duarte JM. Automated evaluation of quaternary structures from protein crystals. PLOS Computational Biology. 2018;14(4):e1006104. doi:10.1371/journal.pcbi.1006104.

41. The HDF Group. Hierarchical Data Format, version 5; 1997-NNNN.

42. Shaikh B, Marupilla G, Wilson M, Blinov ML, Moraru II, Karr JR. RunBioSimulations: an extensible web application that simulates a wide range of computational modeling frameworks, algorithms, and formats. Nucleic Acids Research. 2021;49(W1):W597–W602. doi:10.1093/nar/gkab411.

43. Renaud N, Geng C, Georgievska S, Ambrosetti F, Ridder L, Marzella DF, et al. DeepRank: a deep learning framework for data mining 3D protein-protein interfaces. Nature Communications. 2021;12(1). doi:10.1038/s41467-021-27396-0.

44. Réau M, Renaud N, Xue LC, Bonvin AMJJ. DeepRank-GNN: A Graph Neural Network Framework to Learn Patterns in Protein-Protein Interfaces. bioRxiv. 2021;doi:10.1101/2021.12.08.471762.

45. Freiburger A, Shaikh B, Karr J. BioSimulations: a platform for sharing and reusing biological simulations; 2022. Available from: https://www.hdfgroup.org/2022/02/biosimulations-a-platform-for-sharing-and-reusing-biological-simulations.

46. Berman HM. The Protein Data Bank: a historical perspective. Acta Crystallographica Section A Foundations of Crystallography. 2007;64(1):88–95. doi:10.1107/s0108767307035623.

47. Bourne PE, Berman HM, McMahon B, Watenpaugh KD, Westbrook JD, Fitzgerald PMD. [30] Macromolecular crystallographic information file. In: Methods in Enzymology. Elsevier; 1997. p. 571–590. Available from: https://doi.org/10.1016/s0076-6879(97)77032-0.

48. Bradley AR, Rose AS, Pavelka A, Valasatava Y, Duarte JM, Prlić A, et al. MMTF—An efficient file format for the transmission, visualization, and analysis of macromolecular structures. PLOS Computational Biology. 2017;13(6):e1005575. doi:10.1371/journal.pcbi.1005575.

49. Valasatava Y, Bradley AR, Rose AS, Duarte JM, Prlić A, Rose PW. Towards an efficient compression of 3D coordinates of macromolecular structures. PLOS ONE. 2017;12(3):e0174846. doi:10.1371/journal.pone.0174846.

50. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016;3.

51. Bondi A. van der Waals Volumes and Radii. The Journal of Physical Chemistry. 1964;68(3):441–451. doi:10.1021/j100785a001.

Supplemental Information

Sequence-based bioinformatics tools available in Prop3D

Name	Description	Wikidata entry
BLAST	Search sequences (or groups of sequences) against the non-redundant (NR) database or a custom database	Q286820
DeepSequence	A generative latent variable model for biological sequence families	Q114841036
ESM	Pre-trained language models for proteins	Q114841163
EVcouplings	Evolutionary couplings from protein and RNA sequence alignments	Q114841016
HMMER	Build HMMER models with a group of sequences or at a given level of the CATH hierarchy, search a a group of sequences with a pretrained model, or use jackhmmer starting from a single sequence	Q5631078
MMSeqs2	Ultra-fast and sensitive search and clustering suite	Q114840759
MUSCLE	MUltiple Sequence Comparison by Log-Expectation (state-of-the-art tool for creating MSAs)	Q6719088
SeqDesign	Protein design and variant prediction using autoregressive generative models	Q114841058
USEARCH	High-throughput sequence search and clustering analysis tool	Q114841186

Table 3. Sequence-based bioinformatics tools available in Prop3D. Most of these tools have been dockerized, and are available at our Docker Hub (https://hub.docker.com/u/edraizen).

Structural bioinformatics software suites available in **Prop3D**

Name	Description/purpose (in this context)	Wikidata entry
APBS	Adaptive Poisson-Boltzmann Solver, used here to calculate the electrostatic potential for each atom in a given protein	Q65072984
Consurf	Get pre-calculated conservation scores	Q112888886
CNS	Energy minimize a given structure	Q5191443
CX	Get curvature for each atom in a given protein	Q114841750
DSSP	Calculate secondary structure and accessibility for each residue in a given structure	Q5206192
EPPIC	Calculate sequence conservation scores for a given protein and obtain biologically relevant protein interactions (i.e., not resulting from crystal packing)	Q114841783
foldseek	Fast searching and clustering of protein structure databases	Q114840749
FreeSASA	Get solvent accessibility of each atom in a given protein	Q114841793
Geometricus	A structure-based, alignment-free embedding approach for proteins, utilizing moment invariants	Q114840743
HADDOCK	Dock two proteins or refine the conformation of two docked proteins	Q114841798
MaxCluster	Cluster very similar structures	Q114840623
MGLTools	Convert atom names to AutoDock names and PDBQT	Q114840701
MM-Align	Align two protein complexes	Q114841843
mTM-Align	Multiple structure alignment	Q114841813
MODELLER	Create full atom structures from C_α only models, mutate structures with different amino acids, 'remodel structure' to energy minimize, and model loops	Q3859815
MSMS	Calculate molecular surfaces and create meshes	Q114841806
Multivalue	Merge electrostatic values from multiple atoms, e.g. on a protein surface	Q114840933
OpenBabel	Convert to PDBQT format for AutoDock atom naming	Q612752
PDB2PQR	Protonate a protein structure, debump hydrogens, energy-minimize, and standardise naming (atomic nomenclature)	Q62856803
pdb-tools	A "Swiss army knife of tools" to manipulate PDB files	Q114840802
PRODIGY	Predict binding affinties (K_D) and "fraction of common contacts" in complexes	Q114840854
REDUCE	Protonate and de-protonate structures	Q114840896
SCWRL4	Correct side-chains using the Dunbrack rotamer library	Q114840881
TM-Align	Align two or more protein 3D structures	Q114840775

Table 4. Structural bioinformatics software suites available in Prop3D. Most of these tools have been dockerized, and are available at our Docker Hub (https://hub.docker.com/u/edraizen).

Other software tools used/available in **Prop3D**

Name	Description/purpose (in this context)	Wikidata entry
AlphaFold2	Deep learning-based code for high-accuracy protein 3D structure prediction	Q107711739
AutoDock	A suite of automated protein docking tools	Q4826062
AWS	Amazon Web Services, on-demand cloud computing platforms	Q456157
BioPython	General-purpose collection of open-source tools for computational biology	Q4118434
Biotite	A comprehensive library for computational molecular biology	Q114859551
Docker	Open-source software for deploying containerized applications	Q15206305
HDF5	Hierarchical Data Format, version 5	Q1069215
HSDS	Cloud-native, service-based access to HDF data	Q114859023
h5pyd	Python client library for HDF5 REST interface	Q114859536
Kubernetes	Software to manage containers on a server-cluster	Q22661306
k3s	A light-weight Kubernetes distribution for small servers	Q114860267
MinIO	Cloud storage server compatible with Amazon S3	Q28956397
NumPy	Numerical programming package for the Python programming language	Q197520
Pandas	Python library for data manipulation and analysis	Q15967387
PyTorch	Open-source, Python-based machine learning library	Q47509047
Toil	Enables creation and deployment of massively parallel workflows in Python	Q114858329
Singularity	Open-source container software for scientific environments	Q51294208
SLURM	Free and open-source job scheduler for Linux and similar (Unix-based) operating systems	Q3459703
Oracle Grid Engine	Supercomputer batch-queuing system	Q2708256
Weights and Biases (wandb)	Python library to track machine learning experiments, version data and manage models	Q107382092

Table 5. These other software tools, libraries, etc., for scientific computing and beyond, are either directly used or made available/exposed in the Prop3D ecosystem.

Feature	Voxel Aggregation Rule	Residue Aggregation Rule	Source Software or Database	Wikidata entry
H	max		MGLTools	Q114840701
HD	max		MGLTools	Q114840701
HS	max		MGLTools	Q114840701
C	max		MGLTools	Q114840701
A	max		MGLTools	Q114840701
N	max		MGLTools	Q114840701
NA	max		MGLTools	Q114840701
NS	max		MGLTools	Q114840701
OA	max		MGLTools	Q114840701
OS	max		MGLTools	Q114840701
F	max		MGLTools	Q114840701
MG	max		MGLTools	Q114840701
P	max		MGLTools	Q114840701
SA	max		MGLTools	Q114840701
S	max		MGLTools	Q114840701
CL	max		MGLTools	Q114840701
CA	max		MGLTools	Q114840701
MN	max		MGLTools	Q114840701
FE	max		MGLTools	Q114840701
ZN	max		MGLTools	Q114840701
BR	max		MGLTools	Q114840701
I	max		MGLTools	Q114840701
Unk_atom	max		MGLTools	Q114840701
C_elem	max		PDB File	
N_elem	max		PDB File	
O_elem	max		PDB File	
S_elem	max		PDB File	
H_elem	max		PDB File	
F_elem	max		PDB File	
MG_elem	max		PDB File	
P_elem	max		PDB File	
CL_elem	max		PDB File	
CA_elem	max		PDB File	
MN_elem	max		PDB File	
FE_elem	max		PDB File	
ZN_elem	max		PDB File	
BR_elem	max		PDB File	
I_elem	max		PDB File	
Unk_elem	max		PDB File	
vdw	mean	✓	[51]	
partial charge (charge)	mean	sum	Pdb2Pqr	Q62856803
electrostatic_potential	mean	sum	APBS	Q65072984
concavity (cx)	mean	mean	CX	Q114841750
hydrophobicity	mean	✓	Kyte-Doolittle [35]	
biological_hydrophobicity	mean	✓	[36]	
octanal_hydrophobicity	mean	✓	Wimley-White [37]	
atom_asa	mean		FreeSASA	Q114841793
residue_rasa	mean	✓	DSSP	Q5206192
ALA	max	✓	PDB File	
CYS	max	✓	PDB File	
ASP	max	✓	PDB File	
GLU	max	✓	PDB File	
PHE	max	✓	PDB File	
GLY	max	✓	PDB File	
HIS	max	✓	PDB File	
ILE	max	✓	PDB File	
LYS	max	✓	PDB File	
LEU	max	✓	PDB File	
MET	max	✓	PDB File	
ASN	max	✓	PDB File	
PRO	max	✓	PDB File	
GLN	max	✓	PDB File	
ARG	max	✓	PDB File	
SER	max	✓	PDB File	
THR	max	✓	PDB File	
VAL	max	✓	PDB File	
TRP	max	✓	PDB File	
TYR	max	✓	PDB File	
Unk_residue	max	✓	PDB File	
phi	mean	✓	BioPython	Q4118434
phi_sin	mean	✓	NumPy	
phi_cos	mean	✓	NumPy	
psi	mean	✓	BioPython	Q4118434.
psi_sin	mean	✓	NumPy	
psi_cos	mean	✓	NumPy	
is_helix	max	✓	DSSP	Q5206192
is_sheet	max	✓	DSSP	Q5206192
Unk_SS	max	✓	DSSP	Q5206192
is_regular_helix	max	✓	DSSP	Q5206192
is_beta_bridge	max	✓	DSSP	Q5206192
is_extended_strand	max	✓	DSSP	Q5206192
is_310_helix	max	✓	DSSP	Q5206192
is_pi_helix	max	✓	DSSP	Q5206192
is_hbond_turn	max	✓	DSSP	Q5206192
is_bend	max	✓	DSSP	Q5206192
no_ss	max	✓	DSSP	Q5206192
hydrophobic_atom	max		MGLTools	Q114840701
aromatic_atom	max		MGLTools	Q114840701
hbond_acceptor	max		MGLTools	Q114840701
hbond_donor	max		MGLTools	Q114840701
metal	max		MGLTools	Q114840701
eppic_entropy	min	✓	EPPIC	Q114841783

Table 6. All protein descriptors and extracted features that are defined and calculated in Prop3D. A voxel aggregation method is used to combine two or more atom features if they occupy the same voxels (after accounting for the van der Waals sphere volume). The "Residue Aggregation Rule" describes how the feature is aggregated, from atom to residue, if present in the residue feature. A ✓ indicates if the feature was calculated at the residue level and mapped down to the atom level.

Discretization of continuous feature values in **Prop3D**

Boolean Feature	Source Feature	Equality	Threshold
neg_charge	charge	$<$	0.
pos_charge	charge	$>$	0
is_electronegative	electrostatic_potential	$<$	0.
is_concave	cx	$\leq$	2
is_hydrophobic	hydrophobicity	$>$	0
residue_buried	residue_rasa	$<$	0.2
is_conserved	eppic_entropy	$<$	0.5

Table 7. Boolean operators, and associated threshold criteria listed here, are applied in order to obtain discretized features by converting from continuous values.

How Prop3D abides by the FAIR guidelines

According to the FAIR principles (https://www.go-fair.org/fair-principles), data and code should be Findable, Accessible, Interoperable, and Reproducible. In the following, we enumerate how Prop3D complies with these guidelines:

1. **Findable**

   The first step in (re)using data is to be able to find it. Metadata and data should be easy to find for both humans and computers. Machine-readable metadata are essential for automated discovery of datasets and services, so this is an essential component of the FAIRification process. In Prop3D, the following hold:

   F1. (Meta)data are assigned a globally unique and persistent identifier

   F2. Data are described with rich metadata (defined by R1 below)

   F3. Metadata clearly and explicitly include the identifier of the data they describe

   F4. (Meta)data are registered or indexed in a searchable resource

2. **Accessible**

   Once a user finds the required data, she/he/they need to know how such data can be accessed, possibly including issues of authentication and authorisation. In Prop3D,

   A1. (Meta)data are retrievable by their identifier using a standardised communications protocol

   A1.1 The protocol is open, free, and universally implementable

   A1.2 The protocol allows for an authentication and authorisation procedure, where necessary

   A2. Metadata are accessible, even when the underlying data are no longer available

3. **Interoperable**

   Datasets generally are not used in a vacuum, and at some point or another will need to be integrated with other types and sources of data. In addition, the data need to interoperate with available applications or workflows for analysis, storage, and processing. In Prop3D,

   I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.

   I2. (Meta)data use vocabularies that follow FAIR principles

   I3. (Meta)data include qualified references to other (meta)data

4. **Reusable**

   The ultimate goal of FAIR is to optimise the reuse of data. To achieve this, metadata and data should be well-described so that they can be replicated and/or combined in different settings. In Prop3D,

   R1. (Meta)data are richly described with a plurality of accurate and relevant attributes

   R1.1. (Meta)data are released with a clear and accessible data usage license

   R1.2. (Meta)data are associated with detailed provenance

   R1.3. (Meta)data meet domain-relevant community standards