J Comput Aided Mol Des (2012) 26:387–396
DOI 10.1007/s10822-012-9564-z

A collaborative environment for developing and validating predictive tools for protein biophysical characteristics

Michael A. Johnston · Damien Farrell · Jens Erik Nielsen

Received: 21 December 2011 / Accepted: 18 March 2012 / Published online: 4 April 2012

Abstract The exchange of information between experimentalists and theoreticians is crucial to improving the predictive ability of theoretical methods and hence our understanding of the related biology. However many barriers exist which prevent the flow of information between the two disciplines. Enabling effective collaboration requires that experimentalists can easily apply computational tools to their data, share their data with theoreticians, and that both the experimental data and computational results are accessible to the wider community. We present a prototype collaborative environment for developing and validating predictive tools for protein biophysical characteristics. The environment is built on two central components; a new python-based integration module which allows theoreticians to provide and manage remote access to their programs; and PEATDB, a program for storing and sharing experimental data from protein biophysical characterisation studies. We demonstrate our approach by integrating PEATSA, a web-based service for predicting changes in protein biophysical characteristics, into PEATDB. Furthermore, we illustrate how the resulting environment aids method development using the Potapov dataset of experimentally measured $\Delta\Delta G_{fold}$ values, previously employed to validate and train protein stability prediction algorithms.

Keywords Protein stability · Prediction · Protein design · Data analysis · Data integration · Molecular modelling

Introduction

The interplay between experiment and theory is one of the main forces driving the advance of scientific knowledge. This cross fertilisation is prominent within the fields of structural biology and protein biophysics where theoreticians have made extensive use of on-line experimental databases to aid in the development and validation of predictive models based on protein structure. These include the Protein Data Bank (PDB) [1] and the Catalytic Site Atlas [2] for structural information, ProTherm [3] and BRENDA [4] for stability data, PPD [5] and TitrationDB [6] for pKa values, as well as ligand-affinity databases, for example BindingDB, AffinDB and PDBbind [7–9], and protein–protein interaction databases such as STRING [10]. The resulting models have had success in predicting protein tertiary structures [11, 12] and a host of biophysical properties of proteins (Protein–ligand binding constants [13, 14], pK_a values [15, 16], changes in protein stability [17, 18] and protein–protein interactions [19, 20], and in revealing the mechanisms of fundamental process like catalysis and electron-transfer [21, 22].

The computational tools resulting from these studies have the potential to be of significant benefit to lab-based researchers. Comparing a set of measurements and theoretical predictions of a property can reveal if the underlying theory explains the changes observed. If a strong correlation is found, and if the theoretical model is based on sound physical principles [23], the predictions can be broken down to uncover the specific molecular mechanisms that are causing the observed variation in the protein

Electronic supplementary material The online version of this article (doi:10.1007/s10822-012-9564-z) contains supplementary material, which is available to authorized users.

M. A. Johnston (✉) · D. Farrell · J. E. Nielsen
School of Biomolecular and Biomedical Science, Centre for Synthesis and Chemical Biology, UCD Conway Institute, University College Dublin, Belfield, Dublin 4, Ireland
e-mail: michael.ap.johnston@gmail.com

characteristics. Furthermore the model's predictions can then be used to guide future experiments e.g. suggesting which point mutations would increase binding affinity.

Realising this potential requires deeper and more pervasive collaboration between experimentalists and theoreticians then currently exists. In particular, developing effective tools requires constantly validating them on large amounts of experimental data. This enables identification of outliers, which indicate flaws in a theoretical model and possibly point to new phenomena that the model does not describe adequately. Therefore, it is crucial that comparisons of experimental and predicted values can be captured and made available to theoreticians so they can investigate the reasons for outliers and improve their models.

To facilitate this transfer of information two issues must be addressed. The first is providing experimentalists with easy access to computational methods so they can be continually validated against new experimental datasets. However this is not straightforward for the following reasons:

(1) Many prediction algorithms are computationally intensive. Running such calculations locally can impose a heavy-load on a user's computer and may take days to complete on a single desktop processor.
(2) Users must manually prepare their data, which may involve numerous preprocessing steps. They must also manage the output e.g. by setting up custom spreadsheets for storing computational predictions and comparing them with experimental results.
(3) Often it is time-consuming to update performance metrics e.g. correlation, to reflect changes in the models parameters or corrections/additions to the experimental data. Moreover, computing these quantities using a different theoretical model requires replicating a large number of steps.
(4) Meticulous records of calculation procedures, e.g. input parameters and protein X-ray structures, have to be stored to ensure that other researchers can interpret, redo or expand the predictions.

A common solution to the first of these problems is to provide access to computational tools through web-interfaces. Many examples of this approach exist as witnessed by the ever-expanding number of online services reported in the Nucleic Acids Research yearly web-server issue [24]. Nevertheless, although web-servers remove the need to run calculations locally the other problems remain.

The second issue is a means to easily capture and transfer experimental data to theoreticians and other scientists. This issue is already addressed by our program PEATDB [25] (see section "PEATDB"). PEATDB is an analysis and information system aimed at experimental researchers, that allows for electronic capture, analysis and sharing of experimental data from protein biophysical characterisation experiments.

The challenge is then to provide experimentalists with a way to obtain predictions from remote computational tools from within programs like PEATDB, and to provide developers with a way to quickly set-up and manage access to their tools. The resulting environment would remove most obstacles to collaboration and information transfer, significantly accelerating the process of developing and validating predictive tools for protein biophysical characteristics.

In this article we present a prototype of such a collaborative environment. The base of the environment is PEATDB, while the obstacles outlined above are resolved by two new software components. The first component is a python module, called here RESAT (REmote Service Access Toolkit), which allows developers to provide remote access to their prediction tools and for client programs to interact with them. The second component is PEATDB's plugins facility. A PEATDB plugin can interact with the experimental data stored in the program and also include a simple GUI. To complete our prototype we have used these components to create a plugin which integrates our web-based prediction tool, PEATSA (see section "PEATSA"), into PEATDB.

The PEATSA plugin significantly simplifies experimentalists' interaction with PEATSA. Generation and formatting of the programs inputs is automated, allowing calculations to be submitted with a click of a button. Using RESAT, PEATDB can monitor the progress of a user's calculations and retrieve the results, removing the need for sending emails and checking websites. In addition, since PEATDB understands both the type of prediction the user requested, and the format of PEATSA results, it can provide features that enable the predictions to be visualised, analysed and compared to experimental measurements.

PEATDB's data-sharing capability means experimentalists can share their data and the results of predictions with theoreticians with a click of a button. Furthermore, these results are automatically stored, creating an ever-expanding data-set that theoreticians for tool development. A collaborative environment also makes it straightforward to run new versions of a computational tool on existing data, to compare the performance to previous versions, and to distribute the results for examination to the wider community.

In the next sections we describe the design and implementation of RESAT, covering how it enables communication between the client and web-based server and how it allows access to, and use of, remote resources to be configured. Subsequently we illustrate the advantages of our collaborative environment using the case of developing and validating protein stability prediction tools. Finally, we

discuss how RESAT and PEATDB plugins can be extended to allow other online tools to be accessed from PEATDB. This will provide experimental researchers with the ability to obtain predictions from a host of theoretical models, and also create a platform where the performance of these models can be easily compared.

Methods

PEATDB

PEATDB (Protein Engineering and Analysis Tool—Database)[1] is an application for use in experimental studies of protein biophysical characteristics. It aids in the analysis processing and storage of the large volumes of data that can be produced for a single protein species with current experimental techniques.

Raw experimental data must undergo considerable post-processing to obtain the biophysical quantity of interest. This is often a laborious task involving transforming and fitting the data to physical models. PEATDB removes this burden from the user by providing automated data analysis features that currently include modules for fitting NMR titration curves, thermal melting curves, and enzyme activity assays.

PEATDB also includes extensive data-management and storage features that allow it to manage large amounts of raw experimental data and the data derived from it (i.e. the biophysical characteristics). It also provides data-sharing features that enhance productivity by allowing collaborators to access and work on joint data and to avoid the replication of work. This is a particularly important issue in protein characterisation studies since different research groups often provide separate measurements of multiple characteristics that need to be compared and analysed collectively.

PEATSA

The PEATSA (Protein Engineering and Analysis Tool—Structure Analysis)[2] program provides predictions of the effect of mutations on a protein's stability, ligand-affinity and the pK_a values of its titratable groups. It creates a scalable parallel workflow around a number of other programs including the UFFBAPS empirical force-field [26], pKaTool [27] and pdb2pqr [28]. PEATSA automates many of the tasks associated with setting up such calculations, for example correcting PDB files, adding hydrogens and modelling mutants, and it uses a single output format for the results of all calculation types. These features make it an ideal first candidate for integration with experimental data management software, as none of these tasks have to be implemented on the client side.

The Potapov dataset

The Potapov dataset details 2,154 experimentally measured stability changes caused by single-point mutations (SPMs) in 79 proteins. The experimental data-points were originally extracted from ProTherm [3] and the data-set was first reported in [18]. The Potapov data-set is one of the few datasets which has been used to compare the performance of multiple tools for predicting stability. It also has the advantage that earlier widely used data-sets, such as the Fold-X data-sets [29], are sub-sets of it. Here we have imported those proteins in the Potapov dataset which have five or more SPMs into PEATDB. This gives a set of 1,633 mutations in 42 proteins. The limit of five mutations was imposed since below this it becomes very difficult to identify errors in the predictions due to the protein structure used, or systematic experimental error.

Collaborative environment and use-case setup

This section describes the steps required to configure the collaborative environment and access the use-case data presented in section "Results".

Getting PEATDB

Automatic PEATDB installation packages for Windows, Linux and Mac are available from http://code.google.com/p/peat/downloads/list.

Accessing the Potapov dataset

Once PEATDB is installed and launched users can obtain the Potapov dataset by following these steps:

(1) From the main menu select Project → Remote Project
(2) Enter the following values—Username: guest; Password: 123; Hostname: enzyme.ucd.ie; Port: 8080; Project: PotapovDataset
(3) Click 'OK'

A summary of the dataset will appear in the main window. The "Results" section contains detailed information on how to examine and utilize the dataset.

[1] See http://enzyme.ucd.ie/main/index.php/PEAT_DB.

[2] See http://enzyme.ucd.ie/main/index.php/PEAT_SA and http://enzyme.ucd.ie/PEATSA/.

Configuring PEATSA access in PEATDB

For the purposes of this article we have created a guest account which allows users to access our own PEATSA server (made available via RESAT) and thus run jobs and obtain correlations. Users must first configure PEATDB to access this server. This is done from the PEATSA plugin (Plugins → PEATSA Plugin) by clicking the 'Configure Server' button. The following settings give access to our own cluster: database = DBSAInterface; user = peatdb; password = 123; host = enzyme.ucd.ie.

Advanced options

A number of advanced options for configuring PEATSA jobs are possible and details of these are available at http://enzyme.ucd.ie/main/index.php/PEAT_SA_Plugins. PEATDB also contains many advanced fitting tools and features e.g. outliers can be disabled and enabled and statistical quantities, like correlation coefficients, recalculated. Interested readers are referred to http://enzyme.ucd.ie/main/index.php/PEAT_Data_Fitting for further information.

Implementation

REmote-Service Access Toolkit (RESAT)

The integration of PEATSA and PEATDB is enabled by RESAT, a python module that defines a small set of classes whose state is persisted in a MySQL database. This essentially allows the same object to exist on the client and server machines, providing a means of information transfer between them. Clients use RESAT classes to create entries in the MySQL database representing the calculations ('jobs') they want to run, and to retrieve data from finished jobs. We note that the current implementation of RESAT is influenced by the characteristics of PEATSA, as this was used to develop it. However the structure of the database, and the RESAT API, have been designed from the start to be generic so the module can be extended for use with other services providing predictions of protein biophysical characteristics.

RESAT also includes a program, RESAT Server, which monitors the MySQL database and launches submitted jobs on the remote resource e.g. a cluster. RESAT Server uses the RESAT API to check for new jobs, to access their input data and arguments, and to store the jobs results. Since the API is generic the server contains little PEATSA specific code. Consequently it will be straightforward to adapt it to launch jobs for other programs.

Job submission

The flow of events that occur between job submission and the retrieval of results is illustrated in Fig. 1. This is a UML sequence diagram that shows the main processes involved when a job is submitted, how they communicate with each other and in what order. Five main processes are identified: three of these (PEATDB, PEATSA and RESAT Server) have been described previously. Finally 'Job' and 'JobManager' are the main classes provided by RESAT. The JobManager class is used to create new jobs and provides information on the state of all the jobs in a given MySQL database. Instances of the Job class represent a specific calculation e.g. its configuration, input arguments and state.

A Job can exist in one of four states—Under Construction, Ready, Running and Finished—which define how processes can interact with it. Initially a Job is 'Under Construction'. This is when it is configured and its inputs are set. When a client wants to run a Job it sets its state to 'Ready'. The RESAT server monitoring the database periodically checks for 'Ready' jobs and starts them, at which point their state changes to 'Running'. Once complete their state is finally changed to 'Finished' letting the client know the results of the calculation are available.

Resource allocation

Once computations are moved to the cloud, resource allocation becomes a major concern. Multiple users can simultaneously access the same resources and the question of who can use what, for how long, and when must be resolved. The answers must take into account both fair-share of resources while allowing priority users to get results quickly.

In our implementation we leverage the Torque queuing system and the Maui scheduler to manage these issues [30]. Briefly these tools allow administrators to set up queues which jobs can be submitted to and to define each queue's properties. Some examples of these properties include how many jobs can run at the same time, the priority of jobs in the queue relative to other queues and the compute nodes jobs can be run on.

Users and user-groups can also be defined and configured. For example, each user-group can be assigned a percentage of the available resources e.g. CPUs, and the scheduler will ensure that on average only this amount of resources will be in use by this group at any one time. This guarantees that the jobs submitted by one user-group will not be blocked by another.

Each RESAT Server instance is configured to monitor a single MySQL database and to launch jobs on a specific queue. Thus the database a PEATDB user connects to via the PEATSA plugin defines the set of computational

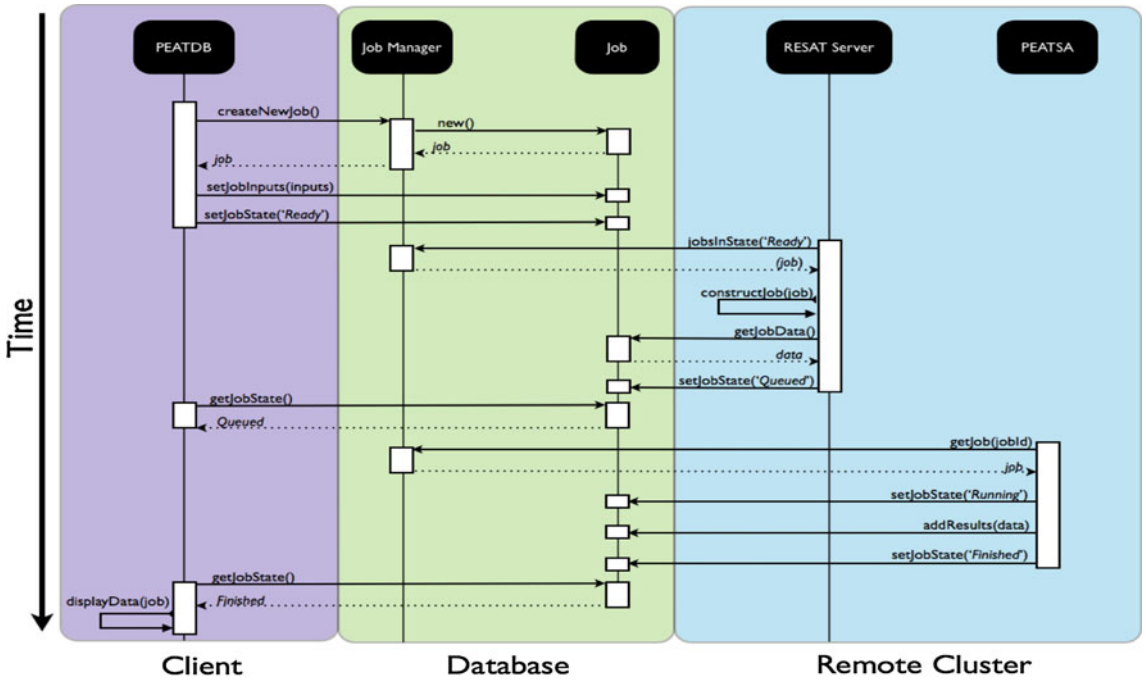

Fig. 1 The Job Submission Process. UML sequence diagram providing a graphical summary of the sequence of events that occurs when a new prediction request (job) is sent from PEATDB. Five programs/objects are shown, the *vertical solid lines* indicating the duration they run for, and the *coloured backgrounds* their location. The *solid* and *dashed arrows* represent the messages sent between these objects and the message results respectively. The *white blocks* denote when the corresponding object is actively processing a message. The sequence of events proceeds from the *top* of the diagram to the *bottom*

resources available to them. This simple implementation enables a great deal of flexibility over how resources are made available. By creating new databases for different types of user the same set of resources can be utilised by a large number of groups. To give a concrete example, our group cluster is utilised by two RESAT servers—one for priority users and another for guests. The jobs submitted to the former have a higher priority and are allocated more resources than those submitted to the latter.

The resources allocated to a group of users can be modified by reconfiguring Torque or Maui or by restarting the server to submit jobs to a different queue. Furthermore, databases do not have to be local to the machine the server is running on. As a result an entire group of users can easily be switched to another set of resources e.g. another cluster, by starting a new RESAT Server which monitors the same database.

Server requirements

Currently RESAT allows anyone to set up a PEATSA service and any python based application can interact with one. Setting up a PEATSA service requires the PEATSA source-code, Torque, MySQL and the MySQLdb python module. The Maui scheduler is optional. All third-party packages are open-source and normally available through the package management program on all Linux distributions.

PEATDB plugins

The graphical component of the PEATSA interface (see Figs. 2, 3) was implemented as a PEATDB plugin. Plugins are python classes that are dynamically loaded at runtime and they provide a lightweight way to add features to the program. A plugin can provide a custom GUI, which is displayed in the main PEATDB window, and they can query and modify the currently open PEATDB project using the PEATDB API. Many of the features of PEATDB are exposed through this API, such as table widgets that allow arbitrary data to be displayed to the user, as well as classes for plotting and fitting data. The PEATSA plugin uses many of these features, examples of which include the calculation status table, the results table, and the correlation plot.

Results

Site-directed mutagenesis of proteins, the substituting of amino-acids in a protein structure with others, has been the most widely used experimental technique for studying

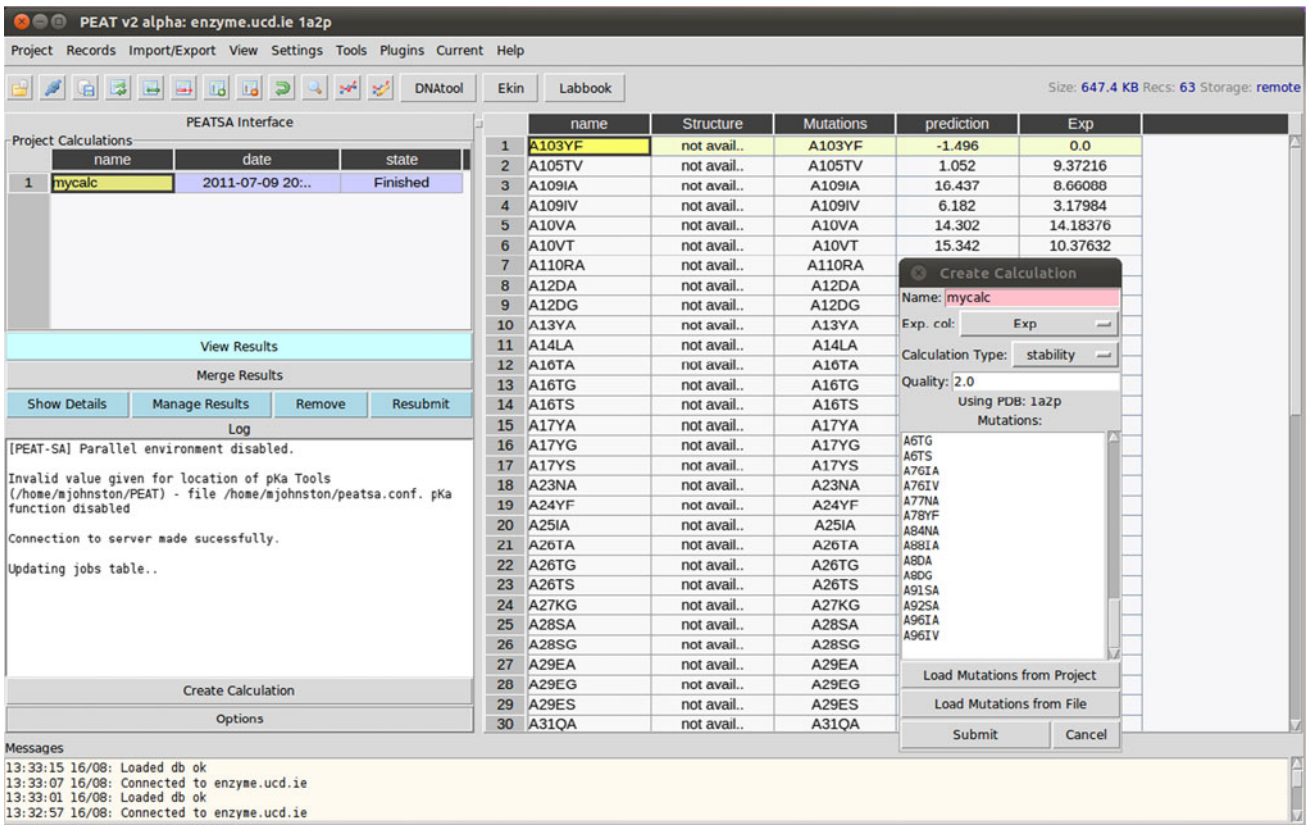

Fig. 2 The PEATSA Plugin Interface. The Barnase project data is on the right side of the window and the PEATSA plugin interface on the left. The mutation codes (present in the 'Mutations' column) have been generated using PDB structure 1A2P. These will automatically update if the structure associated with the project is changed. The 'Project Calculations' table displays the status of all submitted jobs. The 'Create Calculation' dialog allows users to specify the experimental column the predictions are related to, and to define the mutants to test based on the mutation codes in the project. Once 'Submit' is clicked the calculation is sent to the configured remote PEATSA server

protein sequence–structure–function relationships in the last three decades. Furthermore, the development of tools for predicting the effect of single-point mutations (SPMs) on protein stability (termed here SPM Stability Predictors—SPMSPs) has been a highly active area of research in computational biology during the last 10 years [31].

Here we use the example of SPMSP development to demonstrate how a collaborative environment aids experimental investigations and facilitates the validation of theoretical models with experiment. SPMSPs are developed and tested using large sets of experimental data taken from online databases such as ProTherm. However the process of creating such a dataset and validating a program on it, or previously developed sets, is painstaking. In addition there is no current method to share the results of these validation tests. As a result, often the only accessible information on their performance are published statistical measures, such as a correlation coefficient, which can hide many problems [31].

One of the largest such data-sets is the Potapov dataset (see section "The Potapov dataset"). We have imported all proteins in this set with five or more mutations (1,633 total mutations) into PEATDB and used the integration plugin to run PEATSA on them. The data-set and the results are freely accessible through PEATDB. Interested readers can download it, browse the results and correlations, and replicate the steps described here (see section "Collaborative environment and use-case setup" for access details). In the next sections we will describe accessing and browsing the data via PEATDB, how PEATSA predictions are obtained, and how the results can be explored and shared. We have also created a companion screencast (Video S1), which is available as Supplementary Information.

On opening the Potapov project in PEATDB the user is presented with an overview table where each row corresponds to a protein in the data-set. The row contains information such as the number of mutants of that protein in the dataset, along with statistical measures of the performance of the last prediction run on those mutants. The data for each protein is contained in a separate PEATDB project which can be opened by right-clicking on the link in the 'project' column and selecting 'Open Project'.

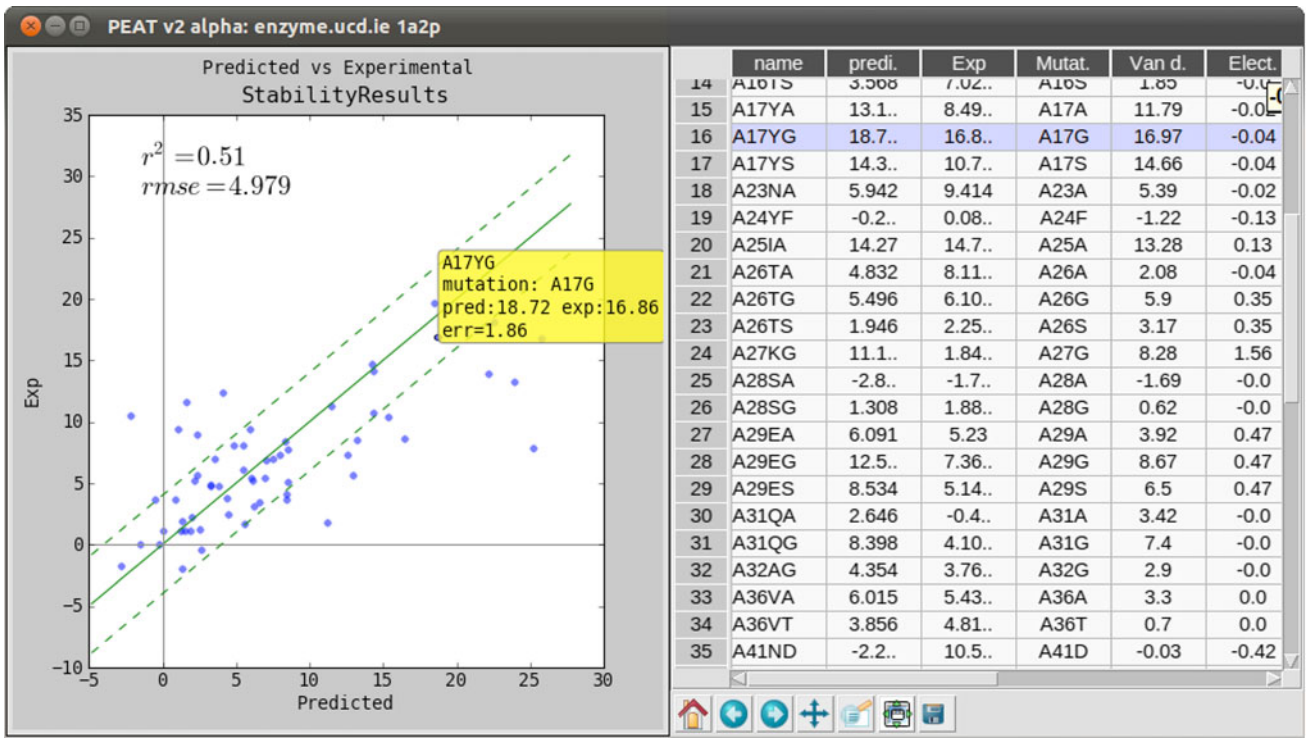

Fig. 3 Comparing Predictions to Experimental Data. Clicking on a point on the correlation plot brings up information including the name of the mutant, the mutation code, the predicted and experimental value and the error. It also *highlights* the corresponding entry in the PEATSA results table allowing the breakdown to be analysed

Case study: Barnase

Here we will examine the results for the protein 1A2P (Barnase) [32–34]. On opening this project the user is presented with a table where each row corresponds to a Barnase mutant (see Fig. 2). In general a PEATDB project can have an arbitrary number of table columns containing information on different properties of the mutants and the results of various experiments. In this case the column labelled 'Exp' contains the experimentally determined change in stability, $\Delta\Delta G_{fold}$, and the column 'prediction' the results of the last PEATSA run, for each mutant (both in kJ/mol). In the following we will obtain the $\Delta\Delta G_{fold}$ predicted by the latest version of PEATSA.

Automatic input preparation

Like all structure-based stability prediction (SP) programs PEATSA requires two inputs. The first is a structure of a protein in PDB format [1] in which the mutations will be modelled. PEATDB users can associate a reference PDB structure with their projects and this structure is the one that is submitted to PEATSA. In the case of Barnase the PDB structure 1A2P has been designated as the reference.

The second are the mutations to model. These are usually supplied as strings, termed here mutation codes. The exact format of a mutation code can vary from program to program but it must uniquely define a set of mutations in a given protein PDB structure. For example in PEATSA the code 'A31R+B1A' describes a mutant where residue 31 in chain A is changed to Arginine and residue 1 in chain B is changed to Alanine.

The main challenge in automatically preparing stability predictor inputs from experimental site-directed mutagenesis data is ensuring that the correct mutations are modelled in the reference PDB structure defined by the user. In a PEATDB project each mutant protein record is associated with an amino acid sequence that has been derived from a wild type sequence. The mutated residues in the protein are thus defined by comparing the residue codes at each position in its amino-acid sequence with the corresponding position in the wild type sequence.

However the residue numbering in the user-supplied PDB structure may not correspond to the numbering in the wild-type sequence. For example, some N-Terminal residues may be missing making residue one in the PDB file equivalent to the third residue in the wild-type sequence. Thus a mutant created by substituting residue one of the wild type must be modelled by substituting residue 3 of the PDB file. Furthermore, the wild type sequence does not contain any chain information and may describe multiple chains e.g. a heterodimer. Therefore we require a way to

identify the chain in a multi-chain PDB file corresponding to the wild-type sequence, and to identify when the wild type sequence maps to two or more PDB chains.

To generate the correct mutation code for a given mutant amino-acid sequence, PEATDB performs a pairwise alignment of the wild-type amino-acid sequence with the chains in the designated reference PDB file [35]. The alignment is used to translate (amino-acid) sequence-specific residue identifiers (e.g. residue 55) into PDB file-specific identifiers (e.g. chain A, residue 45 if the PDB file numbering is offset by 10). The resulting mutation codes are displayed in a column alongside the original data (see Fig. 2).

PEATDB automatically regenerates the PEATSA mutation codes whenever the user changes the reference structure associated with the project. This allows researchers to obtain predictions using different structures of the same protein, either isolated or in complexes with other proteins, without having to worry about adjusting residue numbers and chains ids.

Obtaining predictions

The PEATSA plugin automatically prepares the inputs necessary for a PEATSA calculation. Therefore, once a researcher has defined the reference structure for a project it is straightforward to obtain predictions:

(1) Open the PEATSA plugin interface (Plugins → PEATSA Plugin). A list of previously submitted predictions for the project are displayed in the 'Calculation Summary' table along with their status.
(2) Click 'Create New Calculation'. This launches a dialog for requesting a set of predictions (see Fig. 2).
(3) Give the calculation a name and ensure the 'Experimental Column' is set to 'Stability (kJ/mol)' and the selected calculation is 'Stability'. Note the current PDB structure associated with the project is displayed in the 'PDB' field.
(4) Click on 'Load Mutations from Project'. This causes the plugin to fill the mutation-code input field with all the mutation codes specified in the project. The list can be manually edited if desired.
(5) Finally clicking 'Submit' launches the PEATSA calculation via the remote RESAT server.

The above procedure can be followed by those who have accessed the Potapov dataset, for any of the proteins it contains (note: these steps assume that PEATSA access has been configured as described in section "Collaborative environment and use-case setup"). The time taken for a calculation to complete will depend on the number of mutants submitted, the size of the PDB file and the computational resources available to the RESAT server the calculation is submitted to (see section "Resource allocation").

Examining results

When a calculation is completed, its status changes to 'Finished' in the calculation summary table. Once the predictions are available they can be compared graphically to the experimental results by highlighting the calculation's entry in the table and clicking 'View Results'. If the column containing the reference experimental data was supplied when the calculation was submitted PEATDB will automatically plot the correlation using this data, as shown in Fig. 3. Otherwise the user is prompted for the column to use.

PEATSA predictions are composed of a number of components each quantifying the effect of the mutations on a different physical quantity including Van der Waal's, hydrogen bond, and electrostatic interactions. A table containing these physical quantities is displayed alongside the plot. Clicking on a point on the correlation plot brings up information on the identity of the mutant protein, including the predicted and experimental value, and highlights the associated entry in the component table (see Fig. 3). Each mutant model generated by PEATSA has an associated score, quantifying the model's quality, that can also be examined (see [26] for a detailed description of how this score is calculated).

Sharing results

If desired the PEATSA results can be added to the main project table by highlighting the calculation and clicking 'Merge into Project'. The predictions are matched to the correct sequence entries in the project and appear in a new column alongside the existing data. Once in the main table, researchers can share the predictions with other users who have access to the project by choosing 'Save Changes' from the Project menu. Furthermore, once the data has been merged it can be manipulated as in a standard spreadsheet. For example the predictions can be combined with the results of other calculations to derive the effect of mutations on more complex properties.

Conclusion

In this paper we have described the integration of a remote computational tool, PEATSA, into a platform for storing and sharing experimental data, PEATDB. This integration was achieved using a python module which provides a server for running computational tools on remote resources and an API for interacting with such a server. The result is

a prototype environment which allows experimentalists and theoreticians to collaborate on the development of predictive tools for protein biophysical characteristics.

Although it is difficult to demonstrate the advantages of such an environment before its wide-spread adoption, we believe the examples given here provide ample evidence of the potential benefits. The Barnase case study illustrates how straightforward it becomes for experimentalists to take advantage of computational tools to obtain predictions which can aid their research. Furthermore, it is easy to see that if the experimental data contained in the Potapov dataset had been stored in PEATDB it would have been simple to both assemble, extend, and amend the dataset and to share it with other researchers.

Another example of the uses of this environment can be found in [36]. This was a cross-disciplinary investigation of protein stability during which the methods described in this article were initially developed. Although our solutions could not address all the issues related to such collaborations that we encountered, for example data security, we found that it brought considerable benefits by easing the flow of information between the parties, and by reducing the overhead associated with rerunning predictions.

Beyond cross-disciplinary collaborations our environment also has further benefits for theoreticians. One of the main hurdles in developing new computational tools is providing a means to automatically run them on large standard test sets of experimental data when the programs are improved or when the test dataset is extended. As shown here the integration of predictive tools into PEATDB via RESAT provides an ideal way to handle this problem. Test datasets, like the Potapov set, can be stored in PEATDB and new versions of programs can be automatically run on them. This provides a new level of transparency and cooperation in the development of predictive tools as any researcher can obtain a test dataset, compare the results of running different programs on it, and communicate the results to the method developers. It also aids model development, as it is easy to test new models and spot problematic trends and behaviours that were previously difficult to see.

An example of this second benefit can be seen by examining the per-protein breakdown of the Potapov dataset described here. This reveals that the correlations and standard errors obtained by PEATSA vary considerably from protein to protein, compared to the correlation and standard error for the set as a whole (see [26]). This suggests that some of the data-points in the Potapov dataset may be problematic, since if a trained SPMSP model is generally applicable it should achieve similar correlations and standard errors for all proteins (given enough mutants). These problems could be due, for example, to systematic errors in experimental data, problematic crystal structures or energetic effects that have not been accounted for in the model.

Many other online services related to protein biophysics have similar inputs and outputs to PEATSA, namely a protein structure and results tables, and similar characteristics, for example high computational cost and relatively long job execution time. Such tools are ideal candidates for integration into PEATDB and the resulting plugins could provide the same type of features as described previously for PEATSA. Therefore we are developing our integration module, RESAT, further so it can be used as a general framework for creating client–server interfaces to such programs. This will allow developers to take advantage of the module to provide access to cloud-based computational tools and thus expand the set of predictive methods accessible from PEATDB. Indeed developers will also be able use the module to access such services from their own applications.

For such services RESAT will have advantages over other commonly used methods for accessing remote services such as SOAP or RESTful APIs. It is built specifically for protein-based computational models which eases setting-up a service and interacting with it (via the dedicated API). It also provides several desirable features for such services including integration with a cluster to handle heavy computational loads; automatic storage of runs; automatic handling of security via SQL; and decoupling of the server from the backend where jobs are run.

However these advantages mean RESAT is not a generally applicable solution like SOAP etc. For example, many useful online resources, like databases, do not have the same computational footprint, or outputs, of tools like PEATSA. In this case developers can still exploit PEATDBs plugin architecture (see section "PEATDB plugins"), and the capabilities offered by the PEATDB API, to incorporate these valuable tools into the program using alternative approaches.

Code availability

The python-based integration module utilises MySQL databases and the Torque and Maui (optional) programs. PEATSA, PEATDB and the integration module are licensed under the GPL and the source code is available via http://peat.googlecode.com. In addition this site also provides automatic PEATDB installation package for Windows, Mac and Linux.

Acknowledgments Funding: Science Foundation Ireland (SFI) President of Ireland Young Researcher award (Grant 04/YI1/M537 to J.E.N). SFI Research Frontiers award (Grant 08/RFP/BIC1140 to J.E.N).

References

1. Berman H, Henrick K, Nakamura H (2003) Announcing the worldwide Protein Data Bank. Nat Struct Biol 10(12):980
2. Porter CT, Bartlett GJ, Thornton JM (2004) The Catalytic Site Atlas: a resource of catalytic sites and residues identified in enzymes using structural data. Nucleic Acids Res 32(Database issue):D129–D133
3. Kumar MDS, Bava KA, Gromiha MM, Prabakaran P, Kitajima K, Uedaira H, Sarai A (2006) ProTherm and ProNIT: thermodynamic databases for proteins and protein-nucleic acid interactions. Nucleic Acids Res 34(Database issue):D204–D206
4. Barthelmes J, Ebeling C, Chang A, Schomburg I, Schomburg D (2007) BRENDA, AMENDA and FRENDA: the enzyme information system in 2007. Nucleic Acids Res 35(Database issue): D511–D514
5. Toseland CP, McSparron H, Davies MN, Flower DR (2006) PPD v1.0–an integrated, web-accessible database of experimentally determined protein pKa values. Nucleic Acids Res 34(Database issue):D199–D203
6. Farrell D, Miranda ES, Webb H, Georgi N, Crowley PB, McIntosh LP, Nielsen JE (2010) Titration_DB: storage and analysis of NMR-monitored protein pH titration curves. Proteins 78(4): 843–857
7. Block P, Sotriffer CA, Dramburg I, Klebe G (2006) AffinDB: a freely accessible database of affinities for protein-ligand complexes from the PDB. Nucleic Acids Res 34(Database issue): D522–D526
8. Liu T, Lin Y, Wen X, Jorissen RN, Gilson MK (2007) BindingDB: a web-accessible database of experimentally determined protein-ligand binding affinities. Nucleic Acids Res 35(Database issue):D198–D201
9. Wang R, Fang X, Lu Y, Yang C-Y, Wang S (2005) The PDBbind database: methodologies and updates. J Med Chem 48(12): 4111–4119
10. Jensen LJ, Kuhn M, Stark M, Chaffron S, Creevey C, Muller J, Doerks T, Julien P, Roth A, Simonovic M, Bork P, von Mering C (2009) STRING 8–a global view on proteins and their functional interactions in 630 organisms. Nucleic Acids Res 37(Database issue):D412–D416
11. Rohl CA, Strauss CEM, Misura KMS, Baker D (2004) Protein Structure Prediction Using Rosetta. Methods Enzymol 383:66–93
12. Zhang Y (2008) I-TASSER server for protein 3D structure prediction. BMC Bioinformatics 9:40
13. Sham YY, Chu ZT, Tao H, Warshel A (2000) Examining methods for calculations of binding free energies: LRA, LIE, PDLD-LRA, and PDLD/S-LRA calculations of ligands binding to an HIV protease. Proteins 39(4):393–407
14. Wang R, Lu Y, Wang S (2003) Comparative evaluation of 11 scoring functions for molecular docking. J Med Chem 46(12): 2287–2303
15. Li H, Robertson AD, Jensen JH (2005) Very fast empirical prediction and rationalization of protein pKa values. Proteins 61(4): 704–721
16. Tynan-Connolly BM, Nielsen JE (2006) pKD: re-designing protein pKa values. Nucleic Acids Res 34(Web Server issue):W48–W51
17. Korkegian A, Black ME, Baker D, Stoddard BL (2005) Computational thermostabilization of an enzyme. Science 308(5723): 857–860
18. Potapov V, Cohen M, Schreiber G (2009) Assessing computational methods for predicting protein stability upon mutation: good on average but not in the details. Protein Eng Des Sel 22(9):553–560
19. Aloy P, Russell RB (2004) Ten thousand interactions for the molecular biologist. Nat Biotechnol 22(10):1317–1321
20. Aloy P, Böttcher B, Ceulemans H, Leutwein C, Mellwig C, Fischer S, Gavin A-C, Bork P, Superti-Furga G, Serrano L, Russell RB (2004) Structure-based assembly of protein complexes in yeast. Science 303(5666):2026–2029
21. Olsson MHM, Parson WW, Warshel A (2006) Dynamical contributions to enzyme catalysis: critical tests of a popular hypothesis. Chem Rev 106(5):1737–1756
22. Simonson T (2002) Gaussian fluctuations and linear response in an electron transfer protein. Proc Natl Acad Sci U S A, 99(10):6544–6549
23. Carstensen T, Farrell D, Huang Y, Baker NA, Nielsen JE (2011) On the development of protein pKa calculation algorithms. Proteins. doi:10.1002/prot.23091
24. Benson G (2010) Editorial. Nucleic Acids Res 38(suppl 2): W1–W2
25. Farrell D, O'Meara F, Johnston M, Bradley J, Søndergaard CR, Georgi N, Webb H, Tynan-Connolly BM, Bjarnadottir U, Carstensen T, Nielsen JE (2010) Capturing, sharing and analysing biophysical data from protein engineering and protein characterization studies. Nucleic Acids Res 38(20):e186
26. Johnston MA, Søndergaard CR, Nielsen JE (2011) Integrated prediction of the effect of mutations on multiple protein characteristics. Proteins 79(1):165–178
27. Tynan-Connolly BM, Nielsen JE (2007) Redesigning protein pKa values. Protein Sci 16(2):239–249
28. Dolinsky TJ, Czodrowski P, Li H, Nielsen JE, Jensen JH, Klebe G, Baker NA (2007) PDB2PQR: expanding and upgrading automated preparation of biomolecular structures for molecular simulations. Nucleic Acids Res 35(Web Server issue):W522–W525
29. Guerois G, Nielsen JE, Serrano L (2002) Predicting changes in the stability of proteins and protein complexes: a study of more than 1000 mutations. J Mol Biol 320(2):369–387
30. Bode B, Halstead DM, Kendall R, Lei Z, Jackson D (2000) The portable batch scheduler and the Maui scheduler on Linux clusters. In: ALS'00: Proceedings of the 4th Annual Linux Showcase & Conference. Berkeley, CA, USA: USENIX Association, pp 27–27
31. Johnston MA, Nielsen JE (2011) Constructing and evaluating predictive models for protein biophysical characteristics. Ann Rep Comput Chem 7:101–122. doi:10.1016/B978-0-444-53835-2.00012-2
32. Serrano L, Kellis JT Jr, Cann P, Matouschek A, Fersht AR (1992) The folding of an enzyme. II. Substructure of barnase and the contribution of different interactions to protein stability. J Mol Biol 224(3):783–804
33. Serrano L, Sancho J, Hirshberg M, Fersht AR (1992) Alpha-helix stability in proteins. I. Empirical correlations concerning substitution of side-chains at the N and C-caps and the replacement of alanine by glycine or serine at solvent-exposed surfaces. J Mol Biol 227(2):544–559
34. Horovitz A, Matthews JM, Fersht AR (1992) Alpha-helix stability in proteins. II. Factors that influence stability at an internal position. J Mol Biol 227(2):560–568
35. Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453
36. Farrell D, Webb H, Johnston MA, Poulsen TA, Christensen LB, Borchert TV, Nielsen JE (2012) Towards fast determination of protein stability maps: experimental and theoretical analysis of mutants of a Nocardiopsis prasina serine protease. Biochemistry, Accepted for publication.