Chapter 22

The Bioverse API and Web Application

Michal Guerquin, Jason McDermott, Zach Frazier, and Ram Samudrala

Abstract

The Bioverse is a framework for creating, warehousing and presenting biological information based on hierarchical levels of organisation. The framework is guided by a deeper philosophy of desiring to represent all relationships between all components of biological systems towards the goal of a wholistic picture of organismal biology. Data from various sources are combined into a single repository and a uniform interface is exposed to access it. The power of the approach of the Bioverse is that, due to its inclusive nature, patterns emerge from the acquired data and new predictions are made. The implementation of this repository (beginning with acquisition of source data, processing in a pipeline, and concluding with storage in a relational database) and interfaces to the data contained in it, from a programmatic application interface to a user friendly web application, are discussed.

Key words: Bioverse, framework, systems biology, proteomics, interaction, protein structure, functional annotation, prediction, visualization, server, programming interface, data warehouse.

1. Introduction

The Bioverse project evolved over many years, with the initial idea of a wholistic systems approach to catalogue, predict and present biological information remaining at the heart of the effort. The biological information in the Bioverse is presently specific to the field of 'proteomics'; these include the protein amino acid sequences, known and predicted structures, known and predicted functions and relationships to other proteins such as functional associations in the case of complexes or metabolic pathways and homology.

The Bioverse implementation consists of a pipeline in which this information is transformed and relationships are established, a database where it is organised in an efficient manner (*see* **Chapter 23**),

Jason McDermott et al. (eds.), *Computational Systems Biology, vol. 541*

DOI 10.1007/978-1-59745-243-4_22

an Application Programming Interface (API) that allows specific queries to be issued against the database and a web application that utilises the API to present the data in a browser to Internet users.

In this chapter, we discuss the highly modular Bioverse framework at a conceptual level, detail the implementation of its API and web application components and provide example uses of the framework. We first discuss the data organisation and sources. We then describe the usage of the Bioverse Web Application that is a primary front end to these data. We conclude with an overview of the API and examples of implementing useful programs that use the Bioverse framework.

2. Audience

The Bioverse framework is designed with three audiences in mind: our collaborators who require organism specific information to solve the biological research problems they are working on; bioinformatics experts who are performing large-scale systems analyses of our data; and end users who are seeking detailed information about a particular protein or a small set of proteins. Our collaborators work closely with us and include the Pacific Northwest National Laboratory in Richland, Washington; the National Center for Genetic Engineering and Biotechnology, Thailand (BIOTEC); and the Beijing Genomics Institute who are using our framework for whole genome annotation and comparison *(1, 2)*. Communication with these collaborators occurs through the web site and also by exchange of raw data, so they are the ones likely to obtain the most appropriate results for solving a specific biological problem. The bioinformatics users expect our data to be accessible in a consistent manner and exportable into a format easily transformed into their existing system(s), to the extent that the Bioverse API is developed, they have access to all our data. The end users expect input and output to be simple, easily comprehensible, and to offer rapid insights for specific genes or proteins of interest. Satisfying end users' expectations is a daunting task, since there are many communities with different, ambiguous, and sometimes conflicting desires. In the following sections, we detail how our framework is designed to be valuable to all these user communities.

3. Bioverse Data

3.1. Organisation

Representation of biological systems requires striking a balance between the level of detail and abstraction to solve an intended biological problem. An overly abstract representation will hide the

essential differences between systems, while an overly detailed representation will narrow the scope of the answer to a biological question. Ontologies offer one way of representing relationships between detailed components as concepts in the system. They are therefore one resource to use when presenting related data.

The Bioverse is our first step towards representing the details and organisation of entire biological systems in silico. The Bioverse presently operates on the level of proteins. We organise and describe them in the following ways:

- Individual molecules. These are proteins detected in the sequences of many genomes and characterised in public databases.
- Molecule attributes. Each molecule is uniquely identified by its amino acid sequence. A molecule is assigned one or more names, one or more functional annotations and one or more structure definitions. The latter two may be experimentally determined or predicted in silico. These attributes are in turn described by various meta-data and can be related through organisation methods like the gene ontology (GO) *(3)*.
- Molecule relationships. Relationships between molecules include explicit physical protein–protein interactions, implicit relations such as those present in regulatory complexes and evolutionary relationships based on sequence similarity. Such relations are rich with information when studied in terms of graph theoretic algorithms.
- Collections of molecules. The molecules, or proteins, are associated with an organism they are expressed in.

While the molecules are grouped into sets or collections in organisms, and organisms in turn can be grouped into taxonomic hierarchies, a 'systems biology' perspective encourages thinking that liberally relates many components to one another. This organisation arises from the structure of hierarchies and results in a network of connected components. The components in the Bioverse are currently proteins and some nucleotide sequences but the representational framework is extensible to include DNA, RNA, and biologically important small molecules and ions, and their relationships with each other. The benefits of an inclusive data warehouse become more pronounced as the amount of interrelated data grows. For example, structural and functional genomics projects rely upon the statistical significance of structural and functional feature co-occurrence in large data sets.

3.2. Sources

Studies of individual organisms, or systems in organisms, are being conducted in parallel all around the world. Costly experiments conclude with information about the observed functions or structures of individual proteins or small sets of related proteins. These dispersed pieces of data are meticulously collected into

organism-specific databases such as *Saccharomyces* Genome Database (SGD) *(4)*, WormBase *(5)*, FlyBase *(6)* and Human Protein Reference Database (HPRD) *(7)*. Specific systems-level information is generated in a similar fashion, some focusing on global protein properties like functional annotations, others on catalytic or indirect interactions, and yet others on physical interactions. Here we describe the origins of protein properties, like their names, functional annotations, structures, and interactions as they are catalogued by the Bioverse.

By convention, protein molecules are assigned names. Biologists seeking information about a named molecule can find it in the Bioverse if the protein database where that name occurs has been integrated. The NCBI Gene Identifiers are an example of such names that become searchable name attributes in the Bioverse. There is, of course, the inherent challenge of having multiple naming systems and conventions. Settling on one system simplifies the design but limits the usefulness to users unfamiliar with that system, while adopting many naming systems introduces an excess of data that appears as noise to an uninitiated observer.

Functional annotations are human-readable labels assigned to characterise the behaviour of a protein. Such labels can refer to classes of proteins that are being studied. The landscape of possible functions can be organised into a hierarchy, as demonstrated by GO *(3)*. We use GO for protein functional classification both from predictive methods (such as InterPro *(8)*) and from manually assigned functional annotations from source databases (such as SGD).

Molecule structures, if known, are obtained from RCSB Protein Data Bank (PDB) *(9)*. Classifications of these structures are published by the Structural Classification of Proteins (SCOP) *(10–13)* and Superfamily *(14, 15)* projects, both of which are inherited by the Bioverse. Alternatively, protein structures can be predicted. A prediction of the secondary structure (a positional description indicating sheets, helices and coils) can be obtained by using a program like PSIPRED *(16)*. The three-dimensional tertiary structure can be predicted through various comparative and de novo methods *(17–20)*.

Experimentally observed protein interactions are catalogued in databases like Biomolecular Interaction Network Database (BIND) *(21)*, Human Protein Reference Database (HPRD) *(7)*, Munich Information Center for Protein Sequences (MIPS) *(22)* and Database of Interacting Proteins (DIP) *(23)*, which we use extensively. Sequence similarity computations are performed with BLAST against all molecules in the Bioverse.

These known and computable attributes of molecules form the basis of our predictions. The Interolog method *(24)* is an example of combining sequence similarity information with BIND data to predict novel interactions. Similar methods are applied to function prediction. The specifics of these methods determine our certainty of these predictions and a confidence value is derived accordingly *(25)*.

3.3. Creation and Presentation

The initial data in the Bioverse originates from various sources. After being gathered, it is normalised and collated using a 'pipeline'. The results are stored in a relational database, and a programming interface allows queries to be issued. The Bioverse Web Application utilises this interface and provides a user-friendly interface to the data (**Fig. 22.1**).

3.4. Pipeline

The pipeline distributes data processing across many nodes in a computer cluster *(26)*. These data are organised into a uniform format in preparation for loading into a centralised database, and the execution of algorithms to summarise the data and build

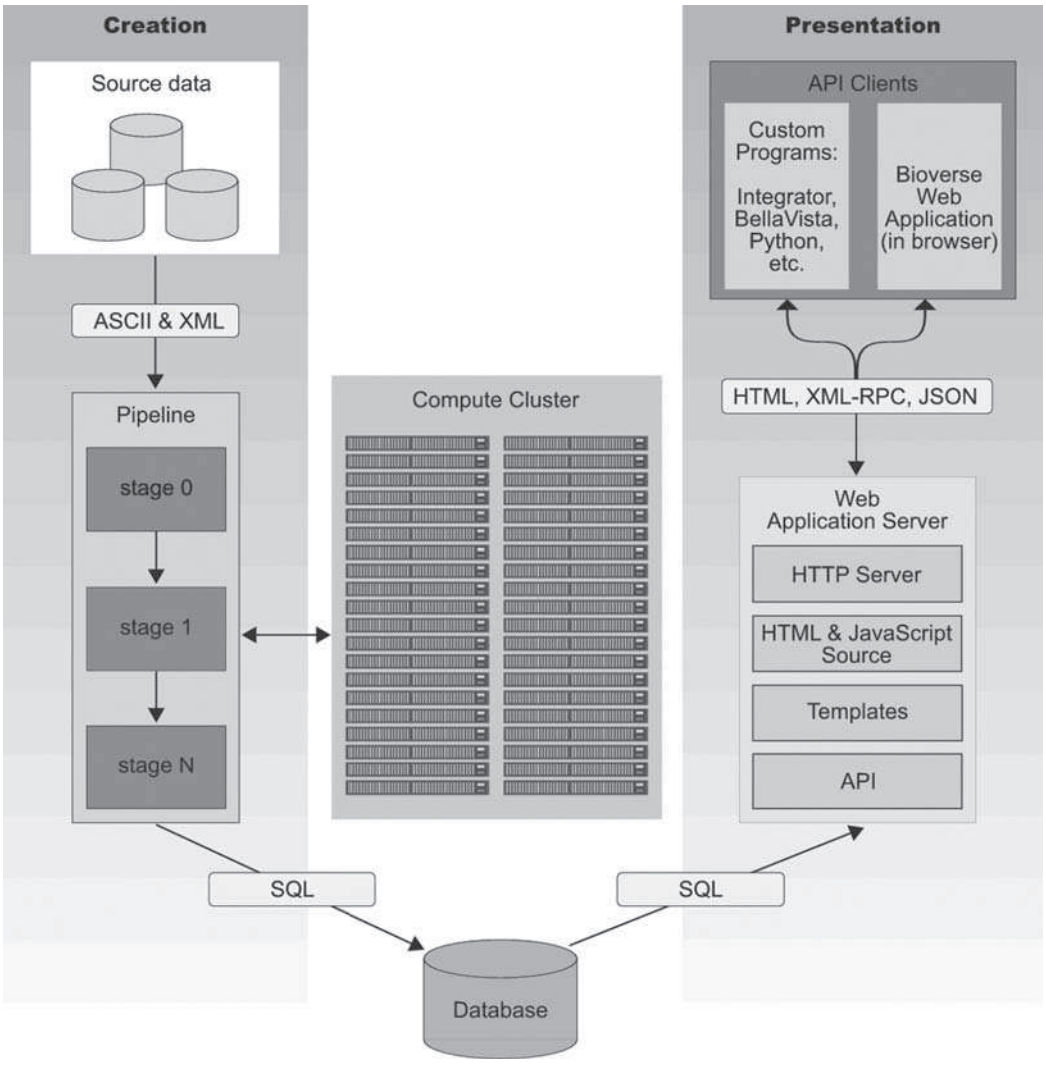

Fig. 22.1. The Bioverse infrastructure described in terms of components and interfaces. External sources are processed in a pipeline and results stored in a database. This database is accessible to the web application server that makes its contents available to the world.

relationships. For example, an all-versus-all profile based PSI-BLAST *(27)* search is executed across all molecule sequences; PSIPRED *(16)* is used for secondary structure prediction and HMMER *(28)* is used to assign proteins to functional families. Novel functional annotations are predicted based on known annotations and the topology of molecule similarity and interaction networks. The pipeline concludes by depositing data into a centralised database (**Table 22.1**).

Table 22.1
The Bioverse holds individual and relationship information for many molecules or proteins. For 54 organisms (486,520 molecules) we require five TB of distributed storage space, two months of dedicated work by 160 CPUs to process and one week to write to a centralised relational database and create indexes on relevant tables. The names, sequences and functional annotations of all molecules are searchable in the Bioverse Web Application and the interaction networks are browsable with Integrator *(42)*

Organism	Molecules	Interaction edges
Agrobacterium tumefaciens (A348 hypothetical)	5,368	5,190
Agrobacterium tumefaciens (C58 Cereon)	5,290	5,067
Agrobacterium tumefaciens (C58 UW)	5,396	4,934
Arabidopsis thaliana	27,833	88,211
Bacillus anthracis (Ames)	5,309	1,965
Bacillus subtilis	4,105	2,556
Bordetella pertussis	3,248	2,690
Brucella melitensis	3,188	2,161
Brucella suis	3,256	2,195
Caenorhabditis elegans	20,936	973,608
Campylobacter jejuni	1,634	1,506
Canis familiaris	16,817	900,459
Chlamydia trachomatis	894	357
Clostridium perfringens	2,722	1,191
Drosophila melanogaster	16,475	5,582,634
Encephalitozoon cuniculi	1,908	6,079
Escherichia coli	4,208	13,196

(continued)

Table 22.1 (continued)

Organism	Molecules	Interaction edges
Halobacterium sp (NRC-1)	2,425	682
Helicobacter pylori (26695)	1,562	9,523
Homo sapiens	26,741	9,362,672
Listeria monocytogenes	2,844	1,588
Magnaporthe grisea	11,042	83,055
Methanococcus jannaschii	1,785	500
Methanococcus maripaludis (C58 UW)	1,722	453
Mus musculus	26,181	10,427,816
Mus musculus (BGI)	12,412	16,135,715
Mycobacterium bovis	3,911	1,812
Mycobacterium tuberculosis (CDC1551)	4,178	1,767
Neisseria meningitidis (mc58)	2,020	1,114
Oryza sativa (indica BGI 9311)	57,135	13,867,984
Oryza sativa (japonica KOME cDNAs)	25,875	699,232
Oryza sativa (japonica Syngenta)	60,017	1,277,025
Pan troglodytes	21,685	3,478,283
Plasmodium falciparum	5,252	29,776
Pseudomonas aeruginosa	5,555	4,977
Pyrococcus abyssi	1,896	64
Pyrococcus furiosus	2,053	834
Rattus norvegicus	22,642	11,810,162
Rhodopseudomonas palustris	4,806	4,556
Rickettsia conorii	1,374	886
Rickettsia prowazekii	834	801
Saccharomyces cerevisiae	5,801	467,989
Salmonella typhimurium	4,532	4,795
Shewanella oneidensis (2a)	4,300	2,908
Shigella flexneri (2a)	4,080	3,983
Staphylococcus aureus (mw2)	2,632	1,385
Thermotoga maritima	1,845	1,031

(continued)

Table 22.1 (continued)

Organism	Molecules	Interaction edges
Vibrio cholerae	3,788	3,249
Vibrio parahaemolyticus	4,821	3,673
Vibrio vulnificus (CMCP6)	4,484	3,925
Yersinia pestis	3,898	4,216
Yersinia pestis (BGI 91001)	4,143	4,448
Yersinia pestis (BGI CO92)	3,708	4,026
Yersinia pestis (BGI KIM)	3,954	4,082
Total	486,520	75,304,986

3.5. Protinfo

The Protinfo suite of servers *(29)*, available at http://protinfo.compbio.washington.edu, comprises several computational techniques for protein structure and function prediction developed by the Samudrala group *(29–41)*. The results of these techniques, as they are applied to all the proteins in the Bioverse, are integrated into the pipeline.

4. Database

The Bioverse stores information in a relational database. Relationships are stored in a space-efficient way to avoid redundancy. In practice, however, we have found the database in a "Write Once, Read Many" pattern of access, so organisational optimisations for purposes of query efficiency are implemented. The nature of this is in the form of a data warehouse. This is discussed in greater detail in **Chapter 23**.

5. Web Application

A traditional web page represents some specialised information with links to other related pages. The intent of a web application, however, is different. Rather than offering small bits of data that are loosely hyperlinked, we cohesively present a large amount of semantically relevant biological information.

The Bioverse Web Application, available at http://bioverse.compbio.washington.edu, is implemented as a single web page. This HTML document is manipulated through JavaScript code executed by the web browser. Interface events, such as button clicks or form submissions, trigger JavaScript functions. These functions may issue calls to the Bioverse API (**Section 6**) to retrieve relevant data and visualise it on the page.

For example, we consider the single molecule view of the Bioverse Web Application (**Fig. 22.2**). The intent of this view is to emphasise the relationship between all information known about a single protein, presented relative to the amino acid sequence of the molecule. For each molecule, this wealth of information is initially presented in a compact form and organised into sections. To expand these sections and see more detailed

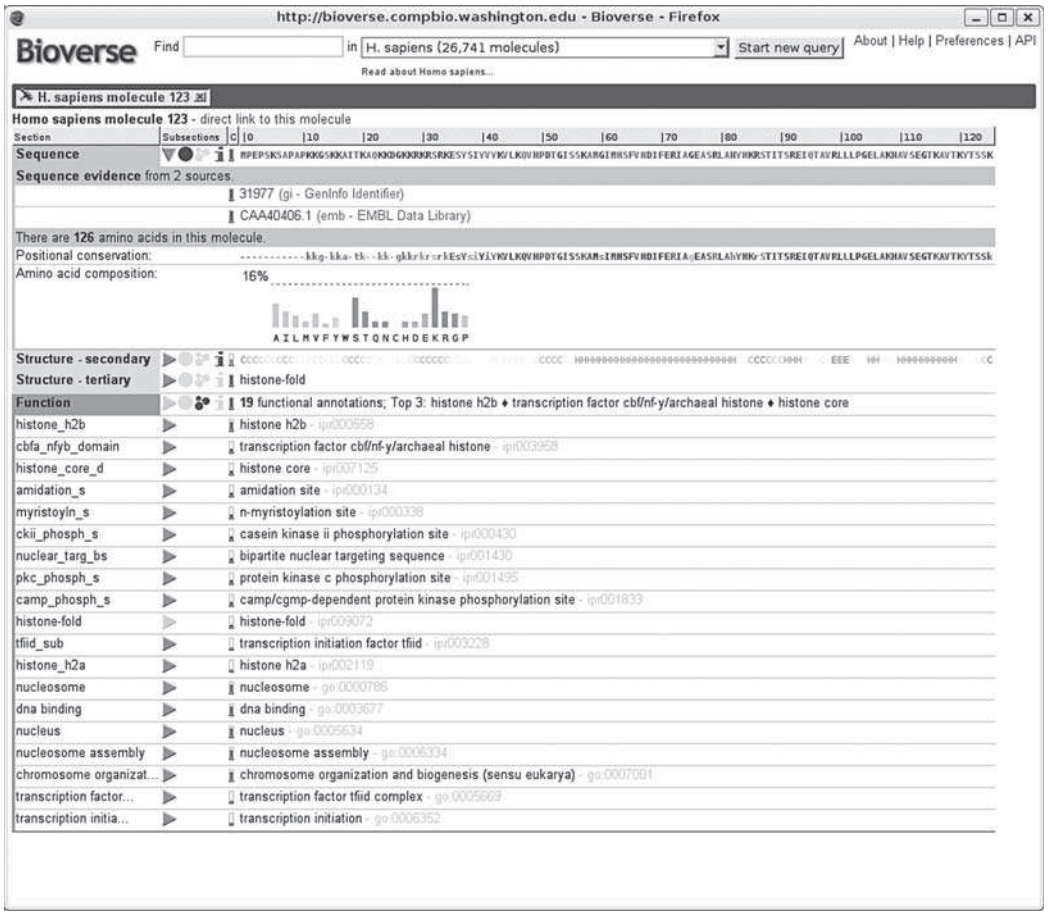

Fig. 22.2. Single molecule view in the Bioverse. The molecule's sequence, structure and function information is shown in three sections. Statistics about sequence composition, as well as amino acid conservation, are expanded. Other sections may be expanded to reveal molecule relationships and evidence supporting predicted functional annotations.

information, buttons can be clicked on the page. Such clicks initiate requests to the server for that information and the page is modified in place by populating the relevant sections only. This reduces the overhead of retrieving increasing quantities of data as more sections are expanded, which occurs in a more traditional web page model.

The traditional meaning of a web page is no longer useful when describing the visible content of a web application. The web browser becomes a platform for executing application logic and rendering content onto an initially empty canvas. Shaping the contents of the page in a piecemeal fashion is a dynamic process and results in tight integration between all the data that can be displayed, and the operations that can be performed on it.

The initial page is empty and contains an onload function call attached to the HTML body element. This function's job is to recover information about which tabs were open from the last session (if any) and to re-open them.

The Bioverse Web Application composes its interface by rendering small sections of content from a pool of predefined HTML templates (**Fig. 22.3**). These are application elements such as the list of organisms, the search form, the search results list, molecule sequence information, and others. The content of these templates is highly specific to their use and templates will often contain placeholders for other templates.

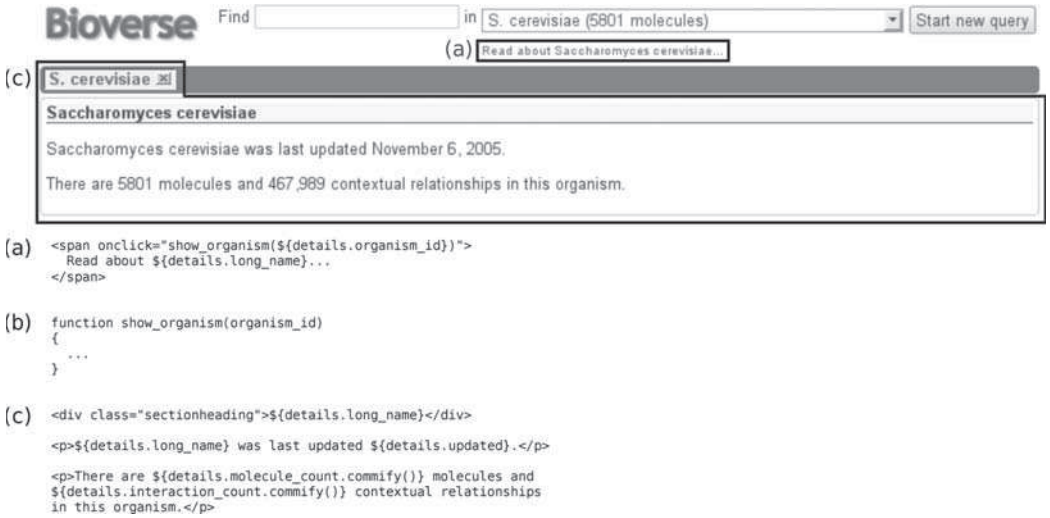

Fig. 22.3. A link on the page (**a**) has an onclick property. When clicked, the browser calls the show_organism function (**b**). This creates a new tab on the page (**c**) with content derived from a template (**d**). Note the use of the template syntax to build the link in (**a**).

In this paradigm of a dynamic web application, the meaning of a "web page" is different. It is no longer possible to accurately represent the state of the web application with a single address. This is resolved by storing a limited subset of state information on the server and recovering it when the web application is loaded. This allows for a browser to follow the traditional model of navigating away from or towards the Bioverse Web Application while relying on the interface initialisation methods in JavaScript to rebuild the application state most recently created by the user.

There are some shortcuts for affecting the state by requesting certain URLs, specifically the referencing of organisms and molecules. For example, the request for the path /about will append the 'About' tab to the set of available tabs in the web application. Other such shortcut URLs are /about/credits, /help, and /preferences. Similarly, visiting /oryza-sativa/123 will add molecule number 123 of *Oryza sativa* (rice) to the set of open tabs.

5.1. Using the Web Application

The hierarchical organisation of most of the data in the Bioverse lends itself well to a 'drill down' presentation where one chooses an organism, then a molecule in the organism and finally the attributes of the molecule to inspect. However, the number of proteins in each organism makes this impractical for the end user. It is therefore necessary to introduce a filtering mechanism, like a search, to focus on a smaller subset of proteins. For example, from a systems perspective, it makes sense to select molecules based on a protein relationship criteria. Searching the data with a free-form user query was inspired by the success of web search engines. The idea is to present the user with a means of writing an expressive description of molecules of interest in a defined syntax and focusing on only those molecules.

The search query is a whitespace separated list of tokens. The tokens may be terms (like binding site, CCR5, GO:000451, MQMRSRMVRLLMML) that can match the name, functional annotation or sequence of a molecule. To restrict which meta-data field of a molecule is searched, the term may be preceded with a meta-data qualifier like name:, function: or sequence:. If a term is not preceded with such a qualifier, then its qualifier is inferred. For example, ABCCCEFH cannot match an amino acid sequence because it contains the letter B, which is not part of the amino acid alphabet. Similarly, dna binding cannot match a molecule name because names cannot contain spaces.

In addition, a token may be preceded with a sign (plus, tilde, or minus character) to indicate the rule (must, may or must not, respectively) for matching molecules (**Fig. 22.4**). A term without a sign indicates that it must occur in the annotation of a molecule. Some example queries are: dna ~binding, +kinase -atp, 'binding

Plus, minus and tilde symbols may precede a term:

+term	means that the term must occur
~term	means that the term may occur
–term	means that the term must not occur

Fig. 22.4. Symbols precede a term to indicate the term's role in the molecule matching algorithm.

site' –function:iron ~rna. This query syntax is sufficiently expressive to allow for quite sophisticated searches to take place. The web interface provides an explanation (**Fig. 22.5**).

Molecules matching these criteria are listed in order of relevance. Relevance is computed by inspecting the confidence value of matching functional annotations and listing molecules with high matching confidence before those with a low matching confidence. Name or sequence matches do not contribute directly to this relevance measure, except to affect the ordering such that molecules with matching functional annotations appear after those matching the name or sequence terms (**Fig. 22.6**).

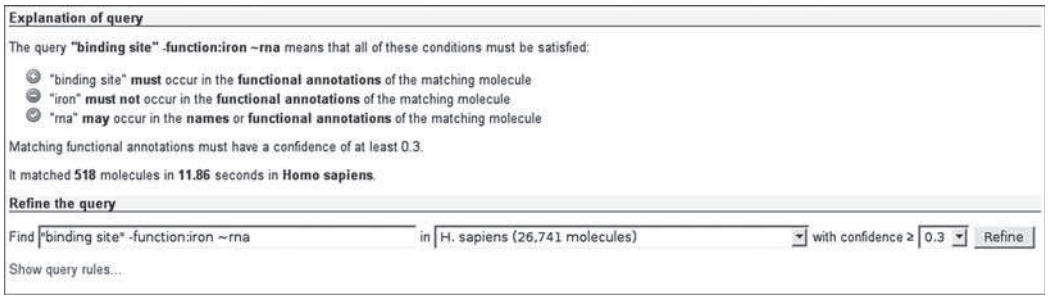

Fig. 22.5. Explanation of a search query. Note that due to ambiguity, the term "rna" may be either a name or a functional annotation.

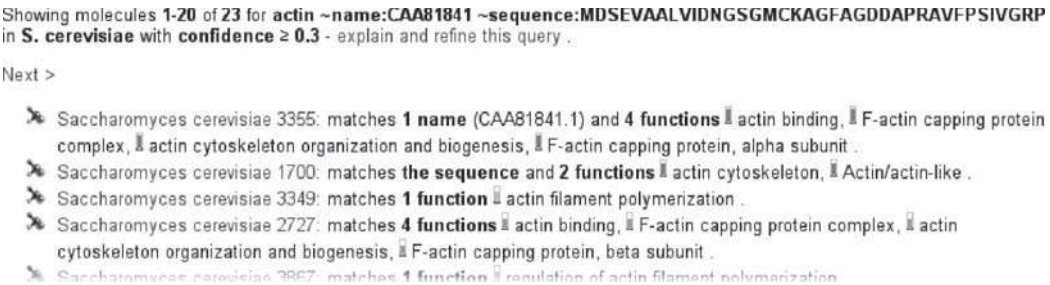

Fig. 22.6. Molecules matching a search query.

Other means of presenting results are being implemented, which involve the algorithmic classification of matching molecules into groups or categories by inspecting shared properties and relationships. This would narrow the focus interactively to a more refined set of molecules, which can be especially useful for exploring result sets that encompass hundreds or thousands of molecules. Superimposing such result sets onto protein interaction networks and visualising these with tools such as Integrator *(42)* can aid in comprehending the structure of the data.

6. Application Programming Interface (API)

The Application Programming Interface (API) is the glue between the data in the database and an application designed to use it. The programming interface is in the form of an Internet service. A programmer with an Internet connection can issue queries using the API and retrieve Bioverse data. In this section we discuss the general idea of the API and devote the next section to describing the API usage.

6.1. The Role of the Bioverse API

Data-intensive web services traditionally create a database and set up a web site to present its contents. To prune the interesting data, a query form exists for a person to fill out, submit to the server and have the results presented. This is ideal for an individual user with questions that can be asked in a predictable way, or for a quick overview of the data in the database.

From the perspective of developing a web site, such a form is actually interfaced to a library of internal server routines that are customised to retrieve the data from the database. These routines are invisible but accessible indirectly through the web form mentioned earlier. We have exposed our library of these internal routines to the world in the form of the Bioverse API.

Applications have already been written to utilise the data in our repository (*see* **Section 6.5**). To emphasise the importance we place upon the API, the most prominent application to utilise it is the Bioverse Web Application itself.

6.2. The Role of the Client

An application that utilises the Bioverse API is considered to be the 'client' of the Bioverse server. The API provides data in a raw but structured format. It is not immediately meaningful for a user, so the application programmer must transform this raw data into a useful form. For example, the amino acid sequence of a protein should have its positions enumerated for visualisation. This is implemented in the Bioverse Web Application by drawing a ruler

Fig. 22.7. (**a** and **b**) Sequence position ruler and confidence bars generated by the client.

above the sequence with positions numbered periodically. Similarly, the confidence of functional annotations *(25)* is a real number between 0 and 1 and can be visualised with coloured images (**Figs. 22.7a and b**).

A notable use of the API by a client application is for the visualisation of relationship networks. The Bioverse API presents fragments of such networks in the form of parent–child relationship lists. It is the role of the client application, such as Integrator *(42)*, to visualise such a list in the form of a graph and accumulate network fragments to 'grow' the graph during exploration (**Figs. 22.8** and **22.9**).

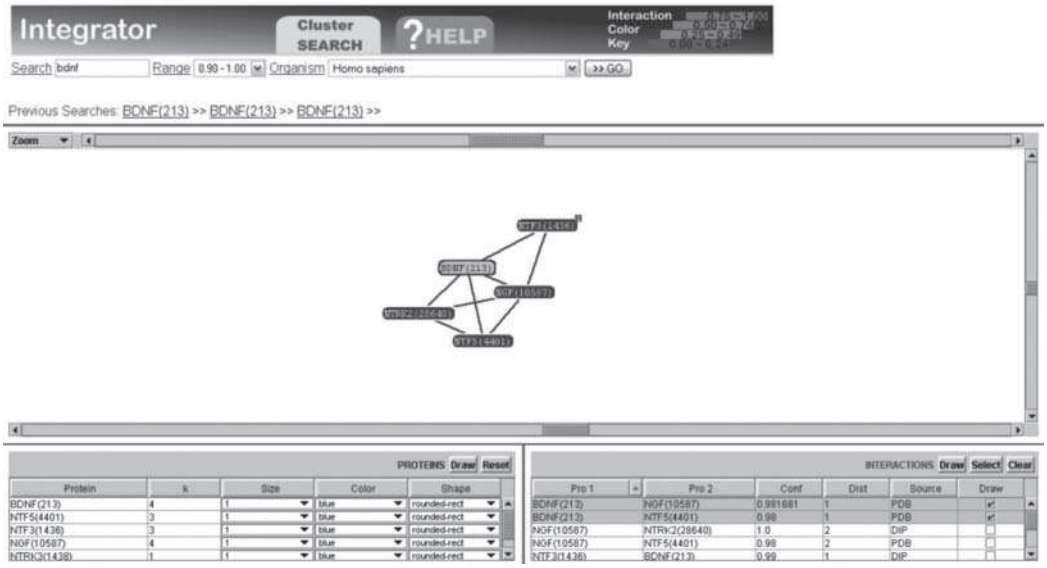

Fig. 22.8. The Bioverse Integrator *(42)* showing a small network of related proteins. The lower interface elements allow the inclusion or exclusion of nodes from the network based on criteria and the adjustment of visual properties like node colours and sizes.

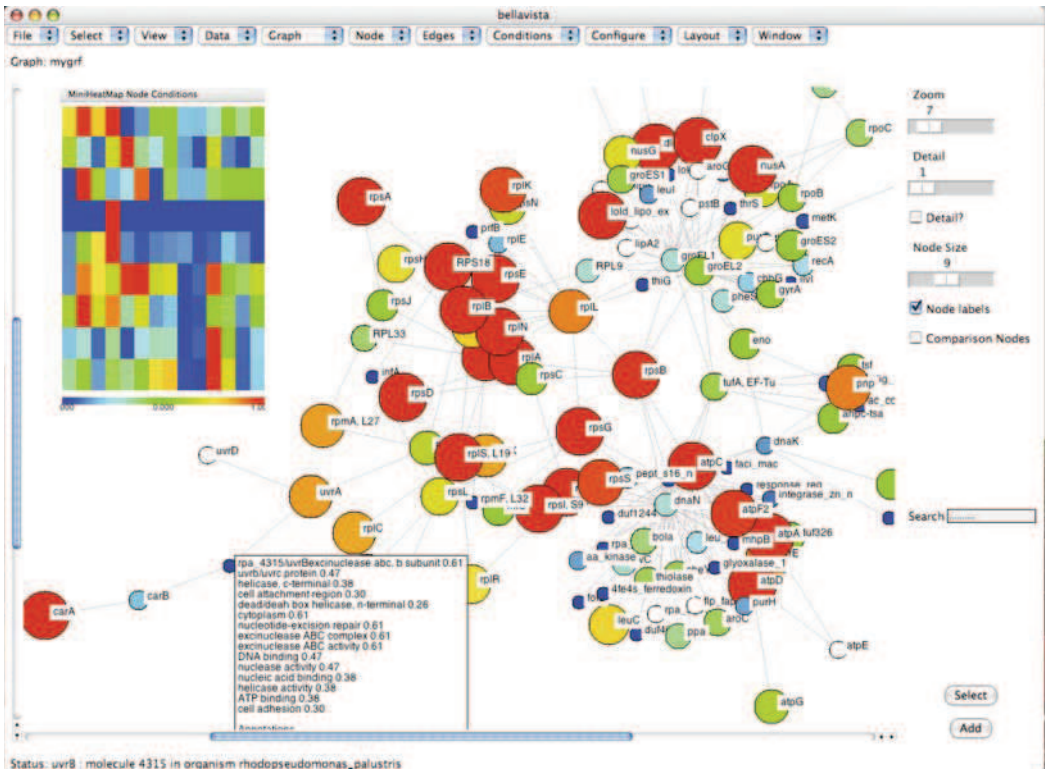

Fig. 22.9. BellaVista *(44)*, a standalone biological information viewer written in Python, visualises proteins, protein properties and protein relationships as nodes in a graph, node attributes and edges between nodes. This flexible application can present information loaded from local files or obtained via the Bioverse API. This includes manual and predicted annotations, protein identifiers, protein similarity relationships (incorporating sources such as the PDB) and Interolog relationships. This screenshot shows a network of protein relationships from *Rhodopseudomonas palustris* overlaid with experimentally obtained proteomics data. Protein abundance levels under different organism growth conditions are integrated into this view from a local file and represented as shades and sizes of nodes. The inset window shows a heat map of members of a putative protein complex under six growth conditions. A pop-up box dynamically lists Bioverse annotations of the selected protein retrieved via the Bioverse API.

6.3. API Usage

The Bioverse Application Programming Interface is a language-agnostic interface to the data and methods implemented in the Bioverse. It allows a programmer to query the Bioverse in many ways. Its capacity is sufficient to allow someone to recreate the Bioverse Web Application in the form of a desktop application.

At the time of writing, there are more than two dozen functions, or methods, publicly available. Because the API is updated periodically, the online documentation should be referenced for the latest information. The examples in the following section are valid at the time of writing, but the API may have changed since then. However, the general idea of accessing the API remains the same. (*See* **Section 6.6** regarding historical changes to the API.)

All methods are available for use by a programmer, but some are intended for use by the Bioverse Web Application and return data that are specific to the way the web application expects it, making them less useful in the general case. Labels, such as general or webapp, assigned to methods in the API documentation hint at the intended usage.

There are presently two ways of accessing the API. One way is via the standardised XML-RPC protocol and the other is via a customised JSON interface.

6.3.1. XML-RPC

The XML-RPC protocol *(43)* hides the complexity of encoding method requests and decoding method responses into and from XML. The result is that a natural syntax can be used to interface with the server. Applications which utilise the XML-RPC protocol include BellaVista *(44)* and the new Integrator code base *(42)*. The API usage examples (**Section 6.5**) utilise the XML-RPC interface.

6.3.2. JSON

JavaScript Object Notation (JSON) *(45)* allows encoding of the same data structures that XML-RPC can encode: numbers, strings, lists, and associative arrays. Unlike XML-RPC, JSON is native to JavaScript and, incidentally, to Python. This convenient confluence makes it a useful dialect of communication between the Bioverse server and the Bioverse Web Application implemented in JavaScript.

The JSON interface is an in-house solution that addresses the problem of the Bioverse Web Application needing to access the same methods as are exposed by the XML-RPC interface. The solution is to encode the request into an HTTP GET query. The server responds to this query with a data structure encoded in the JSON format. That format can be natively and efficiently interpreted by JavaScript.

Incidentally, due to the simplicity of the encoding syntax, it is feasible for a programmer to inspect a JSON response to explore the API or prepare for application implementation. The online API documentation provides example method requests that return JSON data for this purpose (**Section 6.4**).

6.4. API Documentation

The API documentation for each method is formatted for the programmer in a convenient way (**Fig. 22.10**). The three sections of this documentation describe the method's operation and provide usage examples. The examples for the 'Web' section are URLs that return JSON-formatted data. These URLs can be clicked in a browser (to see the JSON-formatted data), embedded into a web application (such as the Bioverse Web Application) or used anywhere the JSON format is convenient. The 'XML-RPC' example code is implemented in Python and serves to illustrate the basic idea of calling the method.

Method reference for molecule_by_name

Labels: general

Documentation:

```
Molecules having specified molecule name

    Input: keys:  molecule_name (string), organism_id (int), db_name (string), limit (int)
    Output: a list of matching molecules and their names

    - molecule_name is required
      - it is used as a substring and checked for containment in the molecule name field
      - it is case insensitive.
    - organism_id and db_name are exclusive.
      - if both given, organism_id overpowers.
      - if neither given, all organisms are considered.
    - limit is optional; if omitted, default is 10.
```

Web example:

```
http://bioverse.compbio.washington.edu/api/web/molecule_by_name?organism_id=7&molecule_name=hox
http://bioverse.compbio.washington.edu/api/web/molecule_by_name?molecule_name=hox
http://bioverse.compbio.washington.edu/api/web/molecule_by_name?db_name=homo_sapiens&molecule_name=H&limit=77
```

XML-RPC example, in Python:

```
import xmlrpclib
server = xmlrpclib.ServerProxy("http://bioverse.compbio.washington.edu/api/xmlrpc/")

print server.molecule_by_name({"organism_id":7, "molecule_name":"hox"})
print server.molecule_by_name({"molecule_name":"hox"})
print server.molecule_by_name({"db_name":"homo_sapiens", "molecule_name":"H", "limit":77})
```

Fig. 22.10. Documentation for the molecule_by_name API method.

6.5. API Usage Examples

Here we provide and explain usage examples currently on the web site. Care should be taken to ensure that the latest API documentation is being referenced. Current examples of using the API are present online at http://bioverse.compbio.washington.edu/api.

In the Python programming environment, accessing the API is trivial. Python will be used throughout this section to illustrate API principles. For example, Program 1 shows how to retrieve a list of all organisms catalogued by the Bioverse.

```
import xmlrpclib
B = xmlrpclib.ServerProxy("http://bioverse.
  compbio.washington.edu/api/xmlrpc/")
print B.organism()
```

Program 1 Listing all organisms catalogued by the Bioverse.

6.5.1. Molecules Annotated by a Single Function

A molecule is annotated by one or more functions. Each function is described by a function identifier (function_id) and a text label. For example, ribulose-phosphate binding barrel is a description of InterPro entry 11060 and is given the Bioverse function identifier of 153787. Program 2 illustrates how we can find all molecules annotated with this function in the organism *Homo sapiens*.

```
(a)
mols = B.molecule_by_function({"organism_id":35,
  "function_id":153787})

(b)
[{"bioverse_id": 64,
 "function_confidence": 0.0,
 "function_desc_1": "RibP_bind_barrel",
 "function_desc_2": "Ribulose-phosphate binding
    barrel",
 "function_id": 153787,
 "function_name": "IPR011060",
 "organism_id": 35},

 {"bioverse_id": 413,
 "function_confidence": 0.25673899999999999,
 "function_desc_1": "RibP_bind_barrel",
 "function_desc_2": "Ribulose-phosphate binding
    barrel",
 "function_id": 153787,
 "function_name": "IPR011060",
 "organism_id": 35},

...
]
```

Program 2 (**a**) Retrieving molecules in the organism *Homo sapiens* (identified by the organism identifier organism_id 35), which are annotated with function identifier 153787 (ribulose-phosphate binding barrel). (**b**) Partial list of resulting dictionaries stored in the variable mols.

The Bioverse function identified by function_id 153787 might have been found earlier with the function_search method, which finds functions that have a description containing a search string. In anticipation of the two-step process of obtaining matching function identifiers and molecules that are annotated with those identifiers, the molecule_by_function method can accept a text string as an argument (like function_search), match it against function descriptions and return molecules that are annotated by those functions (Program 3).

```
mols = B.molecule_by_function(
     {"organism_id":35,
     "function_text":"Ribulose-phosphate binding
       barrel"})
```

Program 3 Alternative version of code in Program 2.

It is important to keep in mind that ribulose-phosphate binding barrel is just a text string with which a simple text search is performed. This means that a search for ribulose will match all

functions containing this string like ribulose-phosphate 3-epimerase, L-ribulose-phosphate 4-epimerase activity, bifunctional ribulose 5-phosphate reductase/CDP-ribitol pyrophosphorylase and, by connection, many more molecules. The string ribulose-phosphate binding barrel was chosen to match only one function in the above example query.

6.5.2. Molecules Annotated by Multiple Functions

In this example we are interested in finding all molecules in *Drosophila melanogaster* (fruit fly), which are annotated by functions containing kinase or phosphate in their descriptions. The function_text search argument of molecule_by_function will be inadequate because of the simple text containment search it performs; it does not accept Boolean expressions. We must instead accumulate a list of functional annotations that match the description kinase or phosphate separately (Program 4).

```
an = []
an += B.function_search({"q":"kinase"})
an += B.function_search({"q":"phosphate"})
```

Program 4 Accumulate functional annotations that contain kinase and phosphate in their description.

Because each function has a unique function_id integer associated with it, which we will need later, we can extract this unique list using standard Python techniques (Program 5).

```
function_ids = list(set([a["function_id"] for a
in an]))
```

Program 5 Get unique list of function identifiers.

To retrieve a list of molecules matching any of these functions, we will use the molecule_by_function method, but now provide it with a list of function identifiers instead of a single value (Program 6). Given this set of molecules, we can print out their various properties (Program 7).

```
mols=B.molecule_by_function({"organism_id":7,
                "function_id":function_ids,
                "limit":0})
```

Program 6 Get all molecules matching functional annotations (as identified in the list function_ids) in *Drosophila melanogaster*, organism 7.

More sophisticated searches can be performed by the method molecule_search, and additional molecule information can be extracted with methods such as molecule_function and molecule_interaction, which are documented online.

```
for m in mols:
  items = [ m["bioverse_id"],
        round(m["function_confidence"],2), #
```

```
            confidence to 2 decimal places
           m["function_name"], # function name like
            GO:12345
           m["function_desc_1"], # function
            description 1
           m["function_desc_2"] # function
            description 2 (if available)
           ]
print "\t".join([str(x) for x in items])
```

Program 7 Display identifiers (bioverse_id) of each molecule in the list mols, and a confidence, a name and two descriptions for each function annotation separated by tabs.

6.6. Versioning of the API Methods

Over time, changes to the API are expected. To keep historical perspective, a list of changes is documented at http://bioverse.compbio.washington.edu/api/versions and old versions of the API remain accessible as long as possible and necessary. This allows for work on an updated version to be underway while maintaining continuity in application behavior.

7. Comparison to Other Similar Projects

The goal of our efforts is a common shared dream among all biologists: to understand how the genome of an organism characterises the development and behaviour of the organism. From a bioinformatics viewpoint, the goal is to organise all the world's biological information to provide semantic meaning through complex models that ultimately model all relationships that occur in life, from atomic level interactions to organismal ones. To this end, several groups have created resources to accomplish goals similar to those outlined here. Some examples include Ensembl *(46)*, Biozon *(47)*, BIND *(21)*, MIPS *(22)*, GRID *(48)*, DIP *(23)*, KEGG *(49)*, 3D-Genomics *(50)*, InterPro *(8)*, PEDANT *(51, 52)*, STRING *(53)*, and Predictome *(54)*. A variety of methods also exist for protein structure, function, and interaction prediction (*see* web server issues of *Nucleic Acids Research*), which can be applied in a large-scale manner to whole proteomes, but in many cases that has not been done or the resulting data are not made available over the web.

In general, annotation databases can be grouped into two categories: those that are both sequence- and structure-oriented and those that are only sequence-oriented. The latter databases

can process larger amounts of data since the amount of structural data is limited, and structural calculations may be time-consuming. Pathway and other network-type databases, such as Predictome *(54)*, are mostly sequence-oriented. Not all databases are comprehensive, i.e. they do not seek to mine data from all completely sequenced genomes simultaneously, or they limit themselves to proteins that are well characterised. Many of the interaction databases are limited to experimentally derived data or manual annotations *(21, 23, 48)*. The databases and software differ in terms of ease-of-use and access to data; some provide bare tables and lists of information whereas others provide some form of abstraction (such as depictions of networks and cellular systems). Few provide programming interfaces, though this trend is changing. There are software-only projects such as Cytoscape *(55)*, which provides a reasonable user interface, but the data for analysis must be explicitly provided to the program instead of referring to a centrally maintained and frequently updated database. Still others that provide predicted interaction information *(56)* are limited to only a few organisms or do not perform novel structural and functional annotation of the interacting proteins *(54)*.

Compared to these projects, the strength of the Bioverse is primarily in our background of developing three-dimensional protein structure and function modelling tools for the past 14 years *(17–20, 29–41)*, which augment the integration of existing data with novel predictions. However, in perspective, all the current and future projects yield complementary information to the scientific community, and it is the synergy of these efforts that is most valuable to the bench biologist seeking to solve a particular domain-specific research problem.

8. Bioverse Technology

All core components of the Bioverse are written in the Python *(57)* programming language running on the Linux operating system. The data warehouse is implemented upon the PostgreSQL *(58)* relational database that resides on a RAID storage array and presently occupies 1.3 terabytes. The web server daemon utilises various free software packages such as CherryPy *(59)* and HTML Templates *(60)*. The web application is written in-house with limited support from external libraries. JavaScript templates *(61)* are used for in-browser content rendering.

Acknowledgements

We acknowledge the invaluable help in the form of comments, contributions, and critiques of the Bioverse from all members of the Samudrala group and the Department of Microbiology at the University of Washington.

Many researchers have helped in the creation of the Bioverse and Protinfo web servers. We thank the scientific community (more properly attributed in **Section 3.2**) for making available data and techniques we have used and relied on.

This work was and is currently supported in part by the University of Washington's Advanced Technology Initiative in Infectious Diseases, Puget Sound Partners in Global Health, NSF CAREER Grant, NSF Grant DBI-0217241, NIH Grant GM068152 and a Searle Scholar Award to Ram Samudrala.

References

1. J. Yu, J. Wang, W. Lin, et al. The genomes of Oryza sativa: a history of duplications. *Public Libr. Sci. Biol. 3*: e38 (2005).
2. S. Kikuchi, K. Satoh, T. Nagata, et al. Collection, mapping, and annotation of over 28,000 cDNA clones from japonica rice. *Science. 301*: 376–379 (2003).
3. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nat. Genet. 25*: 25–29 (2000).
4. J. Cherry, C. Adler, C. Ball, et al. SGD: Saccharomyces genome database. *Nucl. Acids Res. 26*1: 73–79 (1998).
5. T. Harris, N. Chen, F. Cunningham, et al. WormBase: a multi-species resource for nematode biology and genomics. *Nucleic Acids Res. 32*: D411–D417 (2004).
6. F. Consortium. The FlyBase database of the Drosophila genome projects and community literature. *Nucleic Acids Res. 31*: 172–175 (2003).
7. S. Peri, J. D. Navarro, R. Amanchy, et al. Development of human protein reference database as an initial platform for approaching systems biology in humans. *Genome Res. 13*(10): 2363–2371 (2003).
8. R. Apweiler, T. Attwood, A. Bairoch, et al. InterPro-an integrated documentation resource for protein families, domains and functional sites. *Bioinformatics. 16*: 1145–1150 (2000).
9. H. M. Berman, J. Westbrook, Z. Feng, et al. The protein data bank. *Nucl. Acids Res. 28*1: 235–242 (2000).
10. A. G. Murzin, S. E. Brenner, T. Hubbard, C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol. 247*: 536–540 (1995).
11. T. Hubbard, A. Murzin, S. Brenner, C. Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Res. 25*: 236–239 (1997).
12. L. Lo Conte, S. E. Brenner, T. J. P. Hubbard, C. Chothia, A. G. Murzin. SCOP database in 2002: refinements accommodate structural genomics. *Nucl. Acids Res. 30*(1): 264–267 (2002).
13. A. Andreeva, D. Howorth, S. E. Brenner, et al. SCOP database in 2004: refinements integrate structure and sequence family data. *Nucl. Acids Res. 32* (2004).
14. J. Gough, K. Karplus, R. Hughey, C. Chothia. Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. *J. Mol. Biol. 313*: 903–919 (2001).
15. J. Gough, C. Chothia. SUPERFAMILY: HMMs representing all proteins of known structure. SCOP sequence searches,

alignments and genome assignments. *Nucleic Acids Res. 30*: 268–272 (2002).

16. L. McGuffin, K. Bryson, D. Jones. The PSIPRED protein structure prediction server. *Bioinformatics. 16*: 404–405 (2000).
17. R. Samudrala, J. Moult. A graph-theoretic algorithm for comparative modelling of protein structure. *J. Mol. Biol. 279*: 287–302 (1998).
18. R. Samudrala, Y. Xia, E. Huang, M. Levitt. *Ab initio* protein structure prediction using a combined hierarchical approach. *Prot.: Struct. Funct. Genet. S3*: 194–198 (1999).
19. E. Huang, R. Samudrala, J. Ponder. Ab initio fold prediction of small helical proteins using distance geometry and knowledge-based scoring functions. *J. Mol. Biol. 290*: 267–281 (1999).
20. Y. Xia, E. Huang, M. Levitt, R. Samudrala. *Ab initio* construction of protein tertiary structures using a hierarchical approach. *J. Mol. Biol. 300*: 171–185 (2000).
21. G. Bader, D. Betel, C. Hogue. BIND: the biomolecular interaction network database. *Nucleic Acids Res. 31*: 248–250 (2003).
22. H. Mewes, D. Frishman, U. Guldener, et al. MIPS: a database for genomes and protein sequences. *Nucleic Acids Res. 30*: 31–34 (2002).
23. I. Xenarios, L. Salwinski, X. Duan, et al. DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res. 30*: 303–305 (2002).
24. L. Matthews, P. Vaglio, J. Reboul, et al. Identification of potential interaction networks using sequence-based searches for conserved protein-protein intera ctions or "interologs". *Genome Res. 11*: 2120–2126 (2001).
25. J. McDermott, R. Bumgarner, R. Samudrala. Functional annotation from predicted protein interaction networks. *Bioinformatics. 21*: 3217–3226 (2005).
26. Computing. http://compbio.washington.edu/computing.html.
27. S. Altschul, T. Madden, A. Schaffer, et al. Gapped BLAST and PSI-BLAST: a new generation of database programs. *Nucleic Acids Res. 25*: 3389–3402 (1997).
28. HMMER: biosequence analysis using profile hidden Markov models. http://hmmer.janelia.org.
29. L.-H. Hung, R. Samudrala. PROTINFO: secondary and tertiary protein structure prediction. *Nucleic Acids Res. 31*: 3736–3737 (2003).
30. L. Hung, S. Ngan, T. Liu, R. Samudrala. PROTINFO: new algorithms for enhanced protein structure predictions. *Nucleic Acids Res. 33*: W77–W80 (2005).
31. L.-H. Hung, R. Samudrala. An automated assignment-free Bayesian approach for accurately identifying proton contacts from NOESY data. *J. Biomol. NMR. 36*: 189–198 (2006).
32. L.-H. Hung, R. Samudrala. Accurate and automated assignment of secondary structure with PsiCSI. *Protein Sci. 12*: 288–295 (2003).
33. K. Wang, J. A. Horst, G. Cheng, D. C. Nickle, R. Samudrala. Protein Meta-Functional Signatures from Combining Sequence, Structure, Evolution, and Amino Acid Property Information. *PLoS Computational Biology 4*(9): e1000181 (2008).
34. G. Cheng, B. Qian, R. Samudrala, D. Baker. Improvement in protein functional site prediction by distinguishing structural and functional constraints on protein family evolution using computational design. *Nucleic Acids Res. 33*: 5861–5867 (2005).
35. K. Wang, R. Samudrala. FSSA: a novel method for identifying functional signatures from structural alignments. *Bioinformatics. 21*: 2969–2977 (2005).
36. G. Cheng, R. Samudrala. An all-atom geometrical knowledge-based scoring function to predict protein metal ion binding sites, affinities and specificities. *manuscript in preparation* (2007).
37. E. Jenwitheesuk, K. Wang, J. Mittler, R. Samudrala. PIRSpred: a web server for reliable HIV-1 protein-inhibitor resistance/susceptibility prediction. *Trends Microbiol. 13*: 150–151 (2005).
38. E. Jenwitheesuk, R. Samudrala. Prediction of HIV-1 protease inhibitor resistance using a protein-inhibitor flexible docking approach. *Antiv. Ther. 10*: 157–166 (2005).
39. R. Jenwitheesuk, K. Wang, J. Mittler, R. Samudrala. Improved accuracy of HIV-1 genotypic susceptibility interpretation using a consensus approach. *AIDS. 18*: 1858–1859 (2004).
40. K. Wang, E. Jenwitheesuk, R. Samudrala, J. Mittler. Simple linear model provides highly accurate genotypic predictions of HIV-1 drug resistance. *Antiv. Ther. 9*: 343–352 (2004).
41. K. Wang, R. Samudrala. Automated functional classification of experimental and

predicted protein structures. *Bioinformatics. 7*: 278–277 (2006).
42. A. Chang, J. McDermott, Z. Frazier, M. Guerquin, R. Samudrala. INTEGRATOR: interactive graphical search of large protein interactomes over the web. *Bioinformatics. 7*: 146–110 (2006).
43. XML-RPC Home Page. http://www.xmlrpc.com.
44. J. McDermott, M. Guerquin, Z. Frazier, R. Samudrala. BellaVista: a flexible visualization environment for complex biological information. *manuscript in preparation* (2007).
45. JSON. http://www.json.org/.
46. E. Birney, D. Andrews, P. Bevan, et al. Ensembl 2004. *Nucleic Acids Res. 32*: D468–D470 (2004).
47. A. Birkland, G. Yona. BIOZON: a hub of heterogeneous biological data. *Nucl. Acids Res. 34*: D235–D242 (2006).
48. B. Breitkreutz, C. Stark, M. Tyers. The GRID: the general repository for interaction datasets. *Genome Biol. 4*: 744120 (2003).
49. M. Kanehisa, S. Goto, S. Kawashima, A. Nakaya. The KEGG databases at GenomeNet. *Nucleic Acids Res. 30*: 42–46 (2002).
50. K. Fleming, A. Muller, R. MacCallum, M. Sternberg. 3D-GENOMICS: a database to compare structural and functional annotations of proteins between sequenced genomes. *Nucleic Acids Res. 32*: D245–D250 (2004).
51. D. Frishman, M. Mokrejs, D. Kosykh, et al. The PEDANT genome database. *Nucleic Acids Res. 31*: 207–211 (2003).
52. M. L. Riley, T. Schmidt, C. Wagner, H.-W. Mewes, D. Frishman. The PEDANT genome database in 2005. *Nucl. Acids Res. 33*: D308–D310 (2005).
53. C. von Mering, M. Huynen, D. Jaeggi, et al. STRING: a database of predicted functional associations between proteins. *Nucleic Acids Res. 31*: 258–261 (2003).
54. J. Mellor, I. Yanai, K. Clodfelter, J. Mintseris, C. DeLisi. Predictome: a database of putative functional links between proteins. *Nucleic Acids Res. 30*: 306–309 (2002).
55. P. Shannon, A. Markiel, O. Ozier, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res. 13*: 2498–2504 (2003).
56. H. Yu, N. Luscombe, H. Lu, et al. Annotation transfer between genomes: protein-protein interologs and protein-DNA regulogs. *Genome Res. 14*: 1107–1118 (2004).
57. Python Programming Language – Official Website. http://www.python.org.
58. PostgreSQL: The world's most advanced open source database. http://www.postgresql.org.
59. CherryPy. http://www.cherrypy.org.
60. htmltmpl templating engine. http://htmltmpl.sourceforge.net.
61. trimpath – Google Code. http://code.google.com/p/trimpath.