Mapping PDB Chains to UniProtKB Entries

Andrew C.R. Martin*

Department of Biochemistry and Molecular Biology, University College London, Gower Street,
London WC1E 6BT

ABSTRACT

Motivation: UniProtKB/SwissProt is the main resource for detailed annotations of protein sequences. This database provides a jumping-off point to many other resources through the links it provides. Among others, these include other primary databases, secondary databases, the Gene Ontology and OMIM. While a large number of links are provided to PDB files, obtaining a regularly updated mapping between UniProtKB entries and PDB entries at the chain or residue level is not straightforward. In particular, there is no regularly updated resource which allows a UniProtKB/SwissProt entry to be identified for a given residue of a PDB file.

Results: We have created a completely automatically maintained database which maps PDB residues to residues in UniProtKB/SwissProt and UniProtKB/trEMBL entries. The protocol uses links from PDB to UniProtKB, from UniProtKB to PDB and a brute-force sequence scan to resolve PDB chains for which no annotated link is available. Finally the sequences from PDB and UniProtKB are aligned to obtain a residue-level mapping.

Availability: The resource may be queried interactively or downloaded from `http://www.bioinf.org.uk/pdbsws/`

Contact: andrew@bioinf.org.uk –or– a.martin@biochem.ucl.ac.uk

1 INTRODUCTION

The Protein Databank (Berman *et al.*, 2000, PDB) is the primary resource in which protein structure data are deposited while SwissProt and trEMBL (Boeckmann *et al.*, 2003) are the major resources containing protein sequence data. trEMBL is an automatic translation from the EMBL DNA databank while SwissProt contains a huge number of carefully maintained manual annotations. A recent effort has seen the integration of the SwissProt and trEMBL data with the Protein Information Resource (PIR) to create the UniProt Knowledgebase (UniProtKB) (Bairoch *et al.*, 2005), designed as the central access point for extensive curated protein information, including function, classification, and cross-references.

One might expect that mapping between the PDB and UniProtKB would be a trivial exercise. However, this is not the case. In some cases, PDB entries provide cross-links to UniProtKB/SwissProt; in other cases, UniProtKB/SwissProt provides cross-links to PDB; in other cases, no link between the resources is supplied. In the case of cross-links provided from the PDB, this may be to the UniProtKB/SwissProt accession code or the identifier. These may become out-dated and are rarely corrected in PDB entries. Clearly, a mapping between the PDB and UniProtKB/SwissProt is extremely valuable: the detailed annotations and cross-links to other resources available in UniProtKB can then be applied to PDB chains. Recent changes to UniProtKB/SwissProt have improved the mapping and have added PDB chain information and the range of residues in the UniProtKB/SwissProt entry which corresponds to a given PDB chain.

Previously, we developed a mapping between PDB chains and UniProtKB/SwissProt codes as part of an analysis of protein fold distributions for different enzyme classes (Martin *et al.*, 1998). In order to account for supplied links from PDB chain to UniProtKB/SwissProt, UniProtKB/SwissProt entry to PDB file and unlinked files, the process was surprisingly complex. More recently, we developed a system for mapping between PDB protein chains and enzyme classification (EC) numbers, available online at `http://www.bioinf.org.uk/pdbsprotec/` (Martin, 2004) and a system for mapping SNP data onto protein structures (Cavallo and Martin, 2005). In these cases we made use of a mapping between PDB chains and UniProtKB/SwissProt codes made available to us by the EBI. However, the downloadable version of this resource is not regularly updated leading to a need for us to develop an automatically updated version.

Links in the PDB to a UniProtKB/SwissProt entry may contain either the UniProtKB/SwissProt identifier (ID) or the accession code (AC), and are presented at the chain level. Links in the other direction (from UniProtKB/SwissProt to the PDB) are updated more frequently, but until Release 45 of UniProtKB/SwissProt (October, 2004) were at the whole PDB file level. In our previous implementation, `fasta33` (Pearson and Lipman, 1988) was used to resolve which chain was involved and a brute-force FASTA search against UniProtKB/SwissProt was used for any unassigned chains. In 1998, the whole process took some 3–4 days to run and unfortunately was not designed in a manner that could allow fast easy updates as new PDB or UniProtKB/SwissProt entries become available.

With the exponential growth in the size of the PDB and the sequence databanks, a complete run now takes of the order of 10 days on a single Athlon XP 2800+ processor. Since the data resources (UniProtKB/SwissProt, UniProtKB/trEMBL and the PDB) are typically updated within this period, it becomes essential either to parallelize the procedure or, more efficiently, to move to a system which is automatically updateable and does not need to be re-generated from scratch.

The problem of PDB chain to UniProtKB/SwissProt mapping has been partially addressed by the Research Collaboratory for Structural Bioinformatics (RCSB). Their beta site provides a mapping of PDB files to associated UniProtKB/SwissProt entries (`ftp://beta.rcsb.org/pub/pdb/uniformity/derived_data/pdb2sp.txt`). At the time of writing, this file had been updated approximately a month earlier providing UniProtKB/SwissProt accession codes for most PDB chains (some entries, for example 1a7m, are missing). Chimeric chains such as

*To whom correspondence should be addressed chain A of PDB entry 1gk5 (Chamberlin *et al.*, 2001) are also correctly handled, though there is no indication of which residues map to which UniProtKB/SwissProt entry. However, a number of the cross-links appear to be to other databases even when entries appear in UniProtKB/SwissProt. For example, chain N of PDB entry 1a02 is given a cross-link to an entry simply termed ‘1353774’ (presumably an EMBL or Genbank identifier), but this entry should map to UniProtKB/SwissProt entry Q13469. Prior to recent updates, the file had not been updated for almost two years when it had appeared in a different format listing the entries without chain information and providing both UniProtKB/SwissProt identifier and accession for most entries (although the accession was missing from approximately 20% of entries). Clearly the file format is in a state of flux and contains a number of inconsistencies and missing entries where mappings should be possible.

The new XML format files containing PDB data (available at `ftp://beta.rcsb.org/pub/pdb/uniformity/data/XML/`) now provide a mapping between ‘entities’ and UniProtKB/SwissProt entries where an entity corresponds to a unique gene segment. However, this is a beta release; over the course of this work, the format of these files has been in a state of flux and there have been relevant changes. Like the original PDB files, the XML files still suffer from inconsistencies in the use of UniProtKB/SwissProt identifiers and accession codes.

In addition to including cross-links provided in the standard PDB files, the RCSB mapping (in both the flat file and the XML files) includes per-entity mappings derived from cross-links in UniProtKB/SwissProt entries. Thus, while a large amount of information is available, extracting the UniProtKB/SwissProt accession code together with the associated PDB chain and residue range on a regular automated basis remains a complex task. One must either parse all the XML files, or one must use the flat file (assuming it is regularly updated) and resolve cases which are not correct UniProtKB/SwissProt cross-links.

The problem has also been addressed by the Macromolecular Structure Database (MSD) group at the European Bioinformatics Institute (EBI). They have created a mapping between PDB chain and UniProtKB/SwissProt accession code at the individual residue level which is used in the MSDlite search system (`http://www.ebi.ac.uk/msd-srv/msdlite/`). The mapping is not restricted to cross-links found in the two source databases and internally, their mapping correctly handles chimeric chains and provides a complete per-residue mapping. However, this information is not available on the web server version. The data are also available for download (complete with information on chimeras), but the downloadable version is currently only updated on an occasional basis, or on request. While more complete than the mappings provided by the RCSB, it is still incomplete and a number of mappings (particularly those related to viral proteins and antibodies) are absent.

2 METHODS

Here we describe a new, completely automated mapping which we have performed and which is updated automatically as new data become available from the PDB or UniProtKB (containing both UniProtKB/SwissProt and UniProtKB/trEMBL).

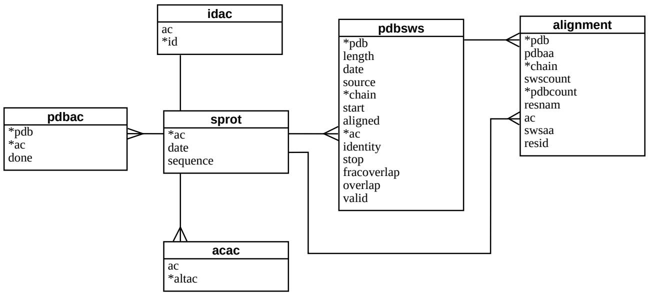

Fig. 1. Entity-relationship diagram for the database used for processing the mapping data. A simple line represents a one-to-one relationship while a ‘crow’s foot’ represents a one-to-many relationship. For example, one UniProtKB accession in the ‘sprot’ table can link to several secondary accessions in the ‘acac’ table. Primary keys are indicated with asterisks. The ‘pdbsws’ table is the main table linking PDB chains to UniProtKB entries while links at the residue level are in the ‘alignment’ table.

The derivation of the mapping occurs in several stages each of which is described below. The mapping is created within a PostgreSQL relational database which may be queried via our web site and is dumped to a flat file which may be downloaded. Each stage of the mapping procedure is designed to be ‘updateable’. In other words, re-running the procedure will only perform the necessary updates. All processing is performed using a set of Perl scripts which interface to PostgreSQL via Perl/DBI.

2.1 Mirroring the data

The UniProtKB data (‘full’ data files incorporating updates on top of official releases) are mirrored from the Expasy FTP site (`ftp://ftp.expasy.org/databases/uniprot/knowledgebase/`) while the PDB files are mirrored from the EBI (`ftp://ftp.ebi.ac.uk/pub/databases/rcsb/pdb/data/structures/all/pdb`). A modified version of the well-known Perl mirror script is used which is capable of storing a local de-compressed version of a remote compressed archive (`http://www.bioinf.org.uk/software/mirror/`).

2.2 Extracting data from UniProt

PostgreSQL (`http://www.postgresql.org/`) is used to store all the data during processing. The overall schema is illustrated in Figure 1. The first processing stage is to extract data from UniProt. A Perl script is run on the UniProtKB/SwissProt data and again on the UniProtKB/trEMBL data. The ‘sprot’ table is populated with the accession code, the sequence and the date of the latest modification. A second ‘acac’ table is populated with mappings between primary and secondary accession codes. A third ‘idac’ table is populated with mappings between identifiers and accession codes. A fourth ‘pdbac’ table is populated with mappings between an accession and a PDB code where these are provided. Since development was started before UniProtKB/SwissProt introduced chain information, these data are currently not stored. For updating, the database table is then scanned for any entries which contain a primary accession code which is now present as a secondary accession code in the ‘acac’ table. These are deprecated entries and are deleted from the database as the data have now been replaced by the new entry.

To allow updating, before an entry is placed in the database, we check whether it exists already. If it does not exist, we proceed as above. If it does exist, then we compare the stored date with the date read from the file. If the date is the same then we simply skip this entry. If the date in the file is newer, then we update the sequence and the date and replace all associated entries in the 'acac' and 'pdbac' tables.

2.3 Extracting chain information and mappings from PDB files

Each PDB file is processed in turn to extract a list of the chains it contains. For each chain, we store the PDB code, the chain, an accession code of '?', a flag to say the entry has not yet been validated, the source of the information ('pdbchain' — simply an indication that this is a chain in the PDB which has not yet been mapped to UniProtKB/SwissProt), the date at which the entry was processed, a flag to say that the entry has not yet been aligned and four zero values which will later be replaced by percentage identity, overlap, length and fraction-overlap when the chain is mapped to a UniProt entry. All these data are stored in a single 'pdbsws' table. This table is the main table which will provide all the mappings from PDB chains to UniProtKB/SwissProt entries.

Cross-references to UniProtKB/SwissProt are also extracted from DBREF records if they are present. These are generally indicated by a database identifier of 'SWS'; however a few cases use 'UNP' to indicate UniProt. If no DBREF records are found then the procedure looks for REMARK 999 records. Where present, these cross-references generally provide one UniProtKB code (ID or accession), but in the case of chimeric chains, there may be more than one code. Thus if there is only one code, we update the existing record in the 'pdbsws' table; if there are more entries, then additional records are inserted into 'pdbsws'. Where an accession code was provided in the PDB file, we change the source of information from 'pdbchain' to 'pdb' indicating that an accession code has been assigned from information in the PDB file.

Currently, we assume that any changes made to a PDB file will not affect the data in which we are interested. Updates to PDB files are rare and tend to be minor in nature. More significant changes such as corrections to the sequence are accommodated by removing a PDB file (making it obsolete), replacing it by a new entry. Therefore updating is handled simply by not re-processing any entry which already exists in the database.

2.4 Fixing PDB cross-references and patching information from UniProtKB for single-chain entries

At this stage we implement a number of corrections to the information extracted from PDB files. First some entries will have provided UniProtKB/SwissProt IDs rather than accession codes. IDs always contain an underscore character, so we replace these entries in the 'pdbsws' table with the accession obtained from the 'idac' table.

A further problem is that while outdated accession codes are maintained in UniProtKB/SwissProt entries as 'secondary accessions', some UniProtKB/SwissProt IDs also change, but no reference to these is maintained in the downloadable UniProtKB/SwissProt files. For example, LYS_CHICK recently changed to LYSC_CHICK. The UniProtKB/SwissProt cross-references in the PDB files are therefore invalid. Such invalid entries are identified and the cross-reference is replaced by an accession of '?'.

During testing, one PDB file was identified which contained both the ID and the accession as separate DBREF records. Since the 'pdbsws' table enforces the constraint that the pdb code, chain and accession must form a unique triplet (a compound primary key), the updates will have failed and we can simply drop the entries for which the code appearing in the 'pdbsws' table is an ID in the 'idac' table and the ID is not the same as the accession. In a few cases, the PDB code appears in the DBREF cross-reference where one expects to find the UniProtKB/SwissProt code. These entries are replaced by accessions of '?'. All such 'problem' entries will be fixed during the brute-force scan.

At this stage, the 'pdbsws' table will contain UniProtKB accession codes for all chains that have cross-references provided in the PDB file even if these had been provided as identifiers. However, some of the accessions may be deprecated secondary accession codes. We therefore replace these with the primary accession code using the data stored in the 'acac' table. If a mapping between a PDB chain and a UniProt primary accession exists, it is possible that the entry with this primary accession will be removed from UniProtKB in a future release. This accession will now become a secondary accession code and a new accession will become the correct primary accession for this PDB chain. The update procedure will automatically correct the accession should this situation arise. It will also ensure that the entry is marked as unaligned, so that the alignment between a PDB chain and the entry will be updated later in the pipeline.

We then validate the accession codes to ensure that they are all valid UniProtKB primary accessions. Once validated, the 'valid' flag in 'pdbsws' table is set.

Some mappings will appear both in PDB files and UniProtKB/SwissProt entries, so we now mark entries in the 'pdbac' table as done wherever we already have a mapping from the PDB. PDB thus takes precedent over UniProtKB/SwissProt because, when development was started, only the mapping in the PDB files was provided at the chain level.

2.5 Adding cross-links from UniProt

In the next stage, we add cross-references supplied by UniProtKB which were not supplied in the PDB files. In the case of single-chain PDB files, this is easy since we can simply transfer the data from the 'pdbac' table to the 'pdbsws' table. The source field is updated to indicate that the information came from UniProtKB/SwissProt.

Since development of this method was started before Release 45 of UniProtKB/SwissProt (the first one to have per-chain mappings to the PDB), the process is more complex in the case of multi-chain PDB files. In this case, we create a file in FASTA format containing the sequences from the PDB file. The UniProtKB sequence is then extracted from the 'sprot' table into a FASTA file and aligned with each sequence in the PDB-derived FASTA file using the Smith-Waterman algorithm as implemented in ssearch33 (Pearson and Lipman, 1988). The best sequence identity is recorded and all chains having this sequence identity are found. Entries in the 'pdbsws' table are updated to link the chain to the UniProt accession and again, the source field is updated to indicate that the information came from UniProt. Future development of the software may make use of the per-chain information from UniProtKB/SwissProt.

For updating, we only transfer accessions from UniProtKB if they do not already exist as valid accessions in the 'pdbsws' table.

In addition, we mark entries in ‘pdbac’ as used once the accession codes have been transferred to the ‘pdbsws’ table. Thus when the processing code is run again, only un-used entries will be considered.

2.6 Brute-force scan

At this stage, all cross-links between PDB files and UniProtKB entries that are available in the source data files have been inserted into the main ‘pdbsws’ table.

Remaining entries in the table will fall into one of the following categories: (1) Chains with entries in UniProtKB which do not have cross-links listed in the source files; (2) Chains which do not have corresponding entries in UniProt; (3) non-protein, or short-peptide chains.

The next stage, therefore, is to perform a ‘brute-force’ scan of remaining PDB chains against the combined UniProtKB (SwissProt and trEMBL) database. First a combined UniProtKB FASTA file is created if either the UniProtKB/SwissProt or UniProtKB/trEMBL FASTA file has been modified since a concatenated version of the two files was last created.

Any PDB chains which have not yet been assigned a valid accession are scanned against the combined UniProtKB data using `fasta33`. The sequence used for the PDB chains is that found in the ATOM records. Any non-standard amino acids are simply ignored (see, for example, the MSE residue at I113, I116 and I182 in PDB entry 487d). While it could be argued that the sequence from the SEQRES records would be more appropriate for this scan, this decision was made for consistency and simplicity since it is the sequence from the ATOM records which must be used in the later alignment stage. The best match is found and is accepted if either (1) the overlap is at least 30 residues and the identity is at least 90%, (2) the overlap is at least 15 residues with at least 93% identity (i.e. 14 out of 15 residues), or (3) the whole of the PDB chain is matched with 100% identity. The ‘pdbsws’ table is updated with the accession code for the best match, together with the percentage identity, the overlap, the length of the PDB chain and the fractional overlap.

Whether or not a match was found, the ‘pdbsws’ table is updated to indicate that the entry has been processed with a brute-force scan and the date on which this was performed is recorded. When running an update, we need to find out whether PDB chains, which previously did not match a UniProtKB entry, match any new or updated entries. Using the date stored for the last processing of an entry, we create a FASTA database containing only those UniProtKB entries that have been added or updated since the PDB chain was last processed and scan using `fasta33` against only those entries.

When updating, we also check PDB chains where the sequence identity was less than 100% or where less than 90% of the PDB chain was matched. This is done to see if there are more recent entries with a better match. Since this is somewhat more time-consuming, it is only done on entries that were last processed at least one month ago.

2.7 Perform alignments

The final stage of the processing is to align PDB chains with UniProt entries. This is slightly inefficient in that alignments may well have been performed already (during the brute-force scan) without storing the alignment data, but it makes the processing much simpler if it is performed as an isolated step. The UniProtKB entry is extracted from the ‘sprot’ table and written in FASTA format. The sequence of the PDB chain is extracted from the ATOM records of the file and also written in FASTA format and the sequences are aligned using `ssearch33`. The alignment is then mapped onto the residue identifiers from the PDB file consisting of the residue number and optional insert code.

Finally the alignments can be dumped to a flat file containing the residue-level mappings between PDB chains and UniProtKB codes.

	Approximate wall-clock time (hours)	
Processing Stage	Initial population	Updating
Processing SwissProt	0.5	0.5
Processing trEMBL	1.5	1.5
Processing PDB files	2.0	0.1
Fixing cross-references, etc	0.5	0.2
Brute Force Scan	216.0	13.0
Performing Alignments	13.5	0.6
Dumping Results	0.3	0.3
Database data analysis	0.5	0.5
Total	234.8	16.7

Table 1. Approximate times required to populate and update the database shown in hours. Timings were on a system using an Athlon XP 2800+ processor, but are highly dependent on other parameters such as disk and network access speeds and, most importantly, the size of the database. ‘Database data analysis’ represents the time taken for PostgreSQL `analyze` steps to update the indexes — see text.

3 RESULTS AND DISCUSSION

Initial population of the database takes approximately 10 days indicating why it is very important that the system is able to be updated. An update corresponding to a new full release of UniProtKB/SwissProt takes less than 17 hours. Approximate timings for populating the database and updating it are shown in Table 1.

The PostgreSQL database is easily able to cope with the rather large tables. The ‘sprot’, ‘idac’ and ‘acac’ tables have more than 2 million rows each, while the ‘alignment’ table contains nearly 8 million rows. However, we found it was important to run the PostgreSQL `analyze` command at regular intervals while populating the database. This updates the statistics on the database contents and allows indexes to work with maximum efficiency. If this was not done, the main ‘postmaster’ process could start to crawl using lots of CPU time and achieving very little.

Table 2 shows the number of chains mapped to UniProt entries from each of the sources of information. The vast majority of entries mapped using a link in the PDB entry will also have a link from UniProt. However, since links from the PDB currently take priority over links from UniProtKB, this information is not recorded.

3.1 Comparison with the EBI mapping

As a validation of the mapping we have created, we have made some comparisons with the mapping produced and kindly provided to us by the EBI.

We have identified one case in which a protein from the wrong species has been identified by our method. PDB entry 1rbf (blank chain name) is an exact match to UniProtKB/SwissProt entry P61824 from *Bison bison*. However 1rbf is a structure of part of

Source of mapping data	Number of chains mapped
PDB entry	40664
UniProtKB	15057[†]
Brute force scan	10324‡
DNA	6261
Short peptides	1647
`fasta33` failed	111
Unmatched	1063

Table 2. Sources of link information in the complete mapping. [†]Since links from PDB to UniProtKB take priority over links in the other direction, this figure considers only those links from UniProtKB to PDB where links in the other direction are absent. ‡While 10324 chains were assigned by the brute-force scan, 815 of these were chains in multi-chain PDB files linked from UniProtKB/SwissProt but which were not identified as matching because other chains matched with a higher sequence identity. The true number of additional chains found by the brute-force scan is therefore 9509.

the chain from *Bos taurus* (P61823). Over the 104 residues of the sequence included in the structure, these two sequences are 100% identical. Chain A of PDB file 1aby (Looker *et al.*, 1992) consists of two copies of the haemoglobin alpha chain (UniProtKB/SwissProt entry P69907) spliced together. Currently our mapping and the EBI MSDLite mapping both match only one of these in the alignment. Thus far, we have identified no other anomalies in our data.

We did, however, find a small number of minor problems in the EBI mapping. PDB entry 1dsj corresponds to UniProtKB/SwissProt entry P12520 and the chain begins with a HETATM 'ACE' group (an N-terminal acetylation) and ends with an additional HETATM 'NH2' group. The most recent downloadable EBI mapping, dated 21st September 2004, maps both of these to real amino acids (Thr49 and Cys76 in the UniProtKB/SwissProt entry respectively). However, the new mapping from UniProtKB/SwissProt to residue ranges within chains has corrected this error.

We also identified an error in the EBI's downloadable mapping for 5azu which contains four identical chains (A–D). All these match UniProtKB/SwissProt entry P00282. However, in their mapping residues 28–30 of the B chain were erroneously identified as coming from Q51325 (this is a secondary accession code for P19919). Again this error does not occur in the mapping from UniProtKB/SwissProt residue ranges to PDB chains.

The mapping provided in the UniProtKB/SwissProt file provides a PDB chain and then specifies the range of residues within the UniProtKB/SwissProt entry that matches that chain. This scheme is unable to address chimeric sequences such as that found in PDB file 1a7m (Hinds *et al.*, 1998). In this PDB file residues 1–47 and 82–180 come from UniProtKB/SwissProt entry P09056 while residues 48–81 come from P15018. In these two UniProtKB/SwissProt entries, a cross-reference to PDB file 1a7m is provided, but the residue range is not given. Our system correctly addresses chimeric chains from the PDB (providing DBREF records are present describing the chimeric construction). The exception to correct processing of chimeric chains is the 'self-chimera', 1aby chain A, described above.

While the downloadable mapping from the EBI is not regularly updated, the MSDLite web server also contains mapping data. We have noted some anomalies in these data as well. For example, while the downloadable mapping for PDB entry 487d adopts the same strategy as us of simply ignoring non-standard amino acids (MSE at I113, I116 and I182), the MSDLite server correctly identifies the UniProtKB entries, but does not include an alignment at all. Similarly for PDB entry 1val, the MSDLite identifies the same UniProtKB entries as our server, but provides no alignment.

At the time of writing, we have identified 115 chimeric chains in the PDB for which residue range mappings are not present in UniProtKB/SwissProt. As shown in Table 2, the brute-force scan of our method identifies approximately 9500 additional chain mappings (representing around 12.5% of chains in the PDB) for which cross-links were not present in either the PDB or UniProtKB/SwissProt. After accounting for DNA chains, short peptides and cases where `fasta33` failed, only around 1050 chains (1.5% of chains in the PDB) were unassigned to UniProt sequences. Some chains, such as antibodies, are only partial assignments: the constant domain is assigned, but the variable domain is not because antibody variable domains do not appear in UniProt.

The procedure also identified a number of errors in the residue ranges specified in `DBREF` records of PDB files. For example, PDB file 1qsn (Rojas *et al.*, 1999) contains a `DBREF` record which indicates that residues 9–19 of chain B should match residues 9–19 of UniProtKB/SwissProt entry P02303 (a secondary accession which has been replaced by P61830). However, the residues in chain B are numbered from 309, so this range should be 309–319. The `DBREF` record in PDB entry 1cxx gives a residue range of 81–193 for the A chain matching Q05158, but the `ATOM` records start from residue 117 and the `SEQRES` records appear to start from 82. Similar problems were identified in PDB entries 1a45, 1dj8, 1dox, 1doy, 1fo7, 1fv2, 1g50, 1g50, 1g6w, 1g6w, 1g6y, 1gd2, 1hgx, 1hqo, 1hqo, 1hr8, 1hr8, 1hr8, 1jid, 1b10, 1k0a, 1k0a, 1k0b, 1k0b, 1ltj, 1m1d, 1kna, 1kne, 4cat, 2pgk, 1bpl.

3.2 Search interface and availability

The complete mapping is available for download via the author's web site at `http://www.bioinf.org.uk/pdbsws/`. The site also provides a search interface allowing searches on the basis of PDB code (optionally with chain label), UniProtKB accession or UniProtKB/SwissProt identifier, all optionally with residue numbers. The results provide links to the Protein Databank and full UniProtKB entries. The web interface also provides a REST-style API (REpresentational State Transfer) — an option to return results in plain text making it easy to parse. This allows simple queries to be made from Perl scripts using the Perl LWP package avoiding the necessity for 'screen scraping' of HTML. This is invaluable for users wishing to employ the results in automated scripts and provides an easy alternative to a SOAP interface. Full instructions are provided on the web site.

4 ACKNOWLEDGEMENTS

The author thanks members of the MSD and SwissProt groups at the EBI (in particular, Sameer Valenka, Virginie Mittard, Phil McNeil, Rolf Apweiler and Kim Henrick) for making their PDB/SwissProt mapping available. This work was funded by a grant from the Wellcome Trust.

REFERENCES

Bairoch, A., Apweiler, R., Wu, C. H., Barker, W. C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., Martin, M. J., Natale, D. A., O'Donovan, C., Redaschi, N. and Yeh, L.-S. L. (2005) The Universal Protein Resource (UniProt), *Nuc. Ac. Res.*, **33**, D154–D159.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. and Bourne, P. E. (2000) The Protein Data Bank, *Nuc. Ac. Res.*, **28**, 235–242.

Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., Martin, M. J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S. and Schneider, M. (2003) The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003, *Nuc. Ac. Res.*, **31**, 365–370.

Cavallo, A. and Martin, A. C. R. (2005) Mapping SNPs to protein sequence and structure data, *Bioinformatics*, **21**, 1443–1450.

Chamberlin, S. G., Brennan, L., Puddicombe, S. M., Davies, D. E. and Turner, D. L. (2001) Solution structure of the mEGF/TGFalpha44-50 chimeric growth factor, *Eur. J. Biochem.*, **268**, 6247–6255.

Hinds, M. G., Maurer, T., Zhang, J. G., Nicola, N. A. and Norton, R. S. (1998) Solution structure of leukemia inhibitory factor, *J. Biol. Chem.*, **273**, 13738–13745.

Looker, D., Abbott-Brown, D., Cozart, P., Durfee, S., Hoffman, S., Mathews, A. J., Miller-Roehrich, J., Shoemaker, S., Trimble, S. and Fermi, G. (1992) A human recombinant haemoglobin designed for use as a blood substitute, *Nature (London)*, **356**, 258–260.

Martin, A. C., Orengo, C. A., Hutchinson, E. G., Jones, S., Karmirantzou, M., Laskowski, R. A., Mitchell, J. B., Taroni, C. and Thornton, J. M. (1998) Protein folds and functions, *Structure*, **6**, 875–884.

Martin, A. C. R. (2004) PDBSprotEC: a Web-accessible database linking PDB chains to EC numbers via SwissProt, *Bioinformatics*, **20**, 986–988.

Pearson, W. R. and Lipman, D. J. (1988) Improved tools for biological sequence comparison, *Proc. Natl. Acad. Sci. USA*, **85**, 2444–2448.

Rojas, J. R., Trievel, R. C., Zhou, J., Mo, Y., Li, X., Berger, S. L., Allis, C. D. and Marmorstein, R. (1999) Structure of Tetrahymena GCN5 bound to coenzyme A and a histone H3 peptide, *Nature (London)*, **401**, 93–98.