PDBrenum: a webserver and program providing Protein Data Bank files renumbered according to their UniProt sequences

Bulat Faezov
Roland L. Dunbrack, Jr

Institute for Cancer Research
Fox Chase Cancer Center
333 Cottman Avenue
Philadelphia PA 19111
USA

February 12, 2021

Abstract

The Protein Data Bank (PDB) was established at Brookhaven National Laboratories in 1971 as an archive for biological macromolecular crystal structures. In the beginning the archive held only seven structures but in early 2021, the database has more than 170,000 structures solved by X-ray crystallography, nuclear magnetic resonance, cryo-electron microscopy, and other methods. Many proteins have been studied under different conditions (e.g., binding partners such as ligands, nucleic acids, or other proteins; mutations and post-translational modifications), thus enabling comparative structure-function studies. However, these studies are made more difficult because authors are allowed by the PDB to number the amino acids in each protein sequence in any manner they wish. This results in the same protein being numbered differently in the available PDB entries. In addition to the coordinates, there are many fields that contain information regarding specific residues in the sequence of each protein in the entry. Here we provide a webserver and Python3 application that fixes the PDB sequence numbering problem by replacing the author numbering with numbering derived from the corresponding UniProt sequences. We obtain this correspondence from the SIFTS database from PDBe. The server and program can take a list of PDB entries and provide renumbered files in mmCIF format and the legacy PDB format for both asymmetric unit files and biological assembly files provided by PDBe. The server can also take a list of UniProt identifiers ("P04637" or "P53_HUMAN") and return the desired files.
Availability: Source code is freely available at https://github.com/Faezov/PDBrenum. The webserver is located at: http://dunbrack3.fccc.edu/PDBrenum.
Contact: bulat.faezov@fccc.edu or roland.dunbrack@fccc.edu.

Introduction

The Protein Data Bank (PDB) is a database for the three-dimensional structural data of biological macromolecules, including proteins and nucleic acids [1]. The data, typically obtained by X-ray crystallography, NMR spectroscopy, or cryo-electron microscopy, and submitted by scientists from around the world, are freely accessible through http://www.wwpdb.org/ and the three World Wide PDB partner sites, https://www.rcsb.org [2], https://www.ebi.ac.uk/pdbe [3], and https://pdbj.org [4]. The PDB provides useful and fundamental information about 1000s of proteins. For many proteins, there are 10s or even 100s of available structures performed under varying conditions, including the presence of different binding partners such as inhibitors, nucleic acids, or other proteins, or with mutations and post-translational modifications. However, in each structure in the PDB, authors are allowed to number protein sequences in any way they wish. This includes the coordinates and any functional or structural annotations contained within the PDB files. Authors commonly number according to sequences deposited in gene databanks such as GenBank [5] or UniProt [6]. So, for instance, a domain from a protein that is not at the N-terminus of the natural sequence may start with its position in the full-length sequence. However, different authors may choose different conventions for this numbering. Authors may or may not include the N-terminal methionine or N-terminal signal sequences in the numbering, both of which may be cleaved off to form the mature protein. For example, in PDB entry 3lvp [7], which is a structure of the kinase domain of human IGF1R, the DFG motif amino acids are numbered 1153-1155. But in PDB entry 3d94 [8], these residues are listed as residues 1123-1125, because the numbering is that of the mature protein, which does not include the 30 amino acid signal sequence. Proteins in the PDB often include N-terminal sequence tags, and the numbering of these residues can be just about anything including negative numbers, 0, or numbers that seem to indicate the residues are from the same gene as the protein under study. These inconsistent numbering schemes compromise structural bioinformatics studies that seek to compare structures within protein families across the PDB. They also affect mapping of sequence annotation data, such as mutations associated with disease, to structural information in the PDB,

since any structure downloaded from the PDB may or may not have the same numbering scheme as the sequence database.

In this paper, we present downloadable computer code in Python3 and a webserver that provide PDB files where the amino acids in all fields are renumbered according to their UniProt sequences. We obtain the correspondence between protein sequences in PDB chains with UniProt entries from the SIFTS database available from PDBe (https://www.ebi.ac.uk/pdbe/docs/sifts/) [9]. Our program and webserver provide renumbered files in mmCIF [10] and legacy PDB format [11] for both asymmetric unit files (the coordinates deposited by authors) from the RCSB server and biological assembly files (provided by the authors or calculated with the program PISA [12]) in mmCIF format from PDBe, which distributes them for all PDB entries. The webserver is easy to use—the user enters a list of PDB codes ("1abc") or UniProt identifiers in the accession code ("P04637") or SwissProt ID ("P53_HUMAN") format, selects which kinds of files to download (mmCIF and/or legacy-PDB format; asymmetric units and/or biological assemblies) and one click enables the download of a zip file containing the requested files.

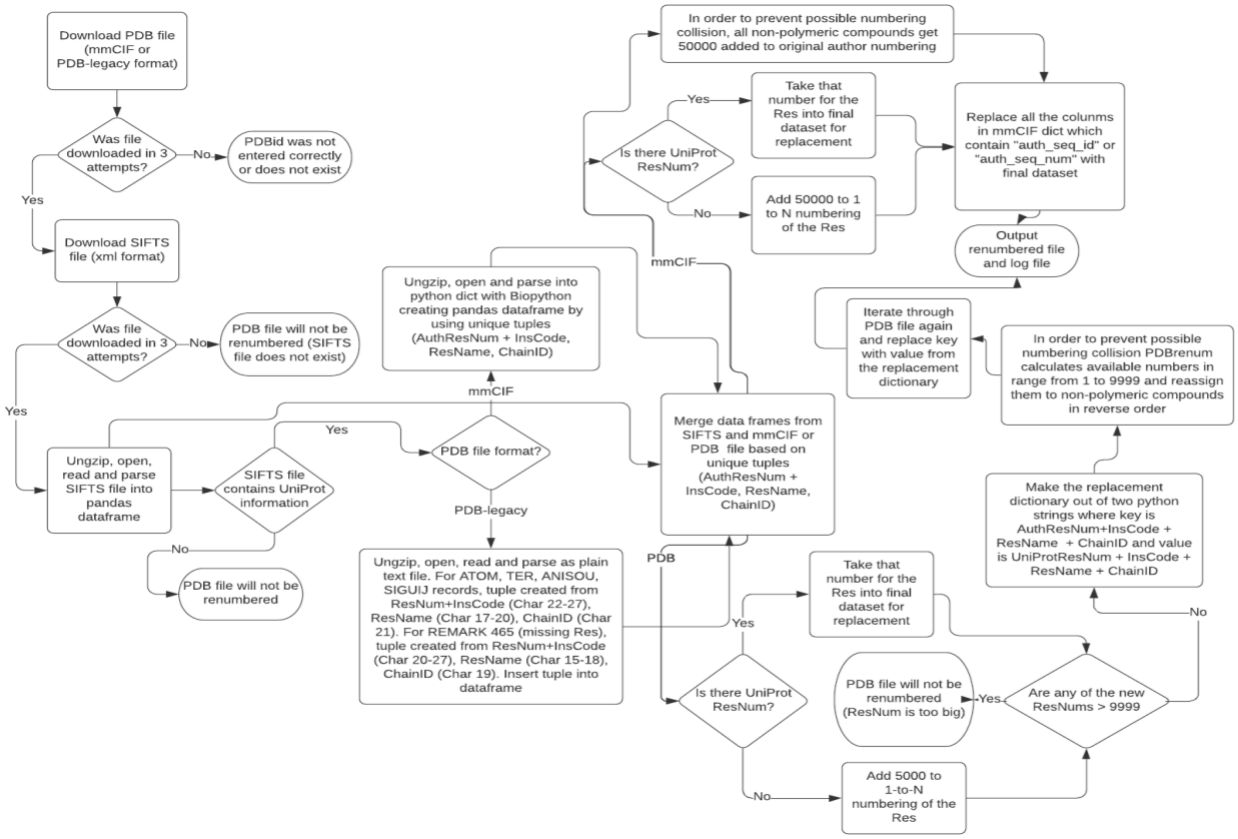

Figure 1. Flow-chart describing basic procedure of PDBrenum

(A)

PDBe	PDB	UniProt	AccessionID	Uni_moD	new_col_Uni	UniProt_50k	Three_Rows_CIF_Num_Uni
(1, ARG, A)	(null, ARG, A)	50000	NaN	(1, ARG, A)	1	50001	[(1, ARG, A), 50001, (1, ARG, A), (null, ARG, ...
(2, GLY, A)	(null, GLY, A)	50000	NaN	(2, GLY, A)	2	50002	[(2, GLY, A), 50002, (2, GLY, A), (null, GLY, ...
(3, SER, A)	(null, SER, A)	50000	NaN	(3, SER, A)	3	50003	[(3, SER, A), 50003, (3, SER, A), (null, SER, ...
(4, HIS, A)	(null, HIS, A)	50000	NaN	(4, HIS, A)	4	50004	[(4, HIS, A), 50004, (4, HIS, A), (null, HIS, ...
(5, HIS, A)	(null, HIS, A)	50000	NaN	(5, HIS, A)	5	50005	[(5, HIS, A), 50005, (5, HIS, A), (null, HIS, ...
(6, HIS, A)	(null, HIS, A)	50000	NaN	(6, HIS, A)	6	50006	[(6, HIS, A), 50006, (6, HIS, A), (null, HIS, ...
(7, HIS, A)	(null, HIS, A)	50000	NaN	(7, HIS, A)	7	50007	[(7, HIS, A), 50007, (7, HIS, A), (null, HIS, ...
(8, HIS, A)	(null, HIS, A)	50000	NaN	(8, HIS, A)	8	50008	[(8, HIS, A), 50008, (8, HIS, A), (null, HIS, ...
(9, HIS, A)	(null, HIS, A)	50000	NaN	(9, HIS, A)	9	50009	[(9, HIS, A), 50009, (9, HIS, A), (null, HIS, ...
(10, GLY, A)	(null, GLY, A)	50000	NaN	(10, GLY, A)	10	50010	[(10, GLY, A), 50010, (10, GLY, A), (null, GLY...
(11, SER, A)	(0, SER, A)	50000	NaN	(11, SER, A)	11	50011	[(11, SER, A), 50011, (11, SER, A), (0, SER, A...
(12, ALA, A)	(1, ALA, A)	(54, A, A)	P30044	(54, A, A)	54	54	[(12, ALA, A), 54, (54, A, A), (1, ALA, A), P3...
(13, PRO, A)	(2, PRO, A)	(55, P, A)	P30044	(55, P, A)	55	55	[(13, PRO, A), 55, (55, P, A), (2, PRO, A), P3...
(14, ILE, A)	(3, ILE, A)	(56, I, A)	P30044	(56, I, A)	56	56	[(14, ILE, A), 56, (56, I, A), (3, ILE, A), P3...
(15, LYS, A)	(4, LYS, A)	(57, K, A)	P30044	(57, K, A)	57	57	[(15, LYS, A), 57, (57, K, A), (4, LYS, A), P3...
(16, VAL, A)	(5, VAL, A)	(58, V, A)	P30044	(58, V, A)	58	58	[(16, VAL, A), 58, (58, V, A), (5, VAL, A), P3...
(17, GLY, A)	(6, GLY, A)	(59, G, A)	P30044	(59, G, A)	59	59	[(17, GLY, A), 59, (59, G, A), (6, GLY, A), P3...
(18, ASP, A)	(7, ASP, A)	(60, D, A)	P30044	(60, D, A)	60	60	[(18, ASP, A), 60, (60, D, A), (7, ASP, A), P3...
(19, ALA, A)	(8, ALA, A)	(61, A, A)	P30044	(61, A, A)	61	61	[(19, ALA, A), 61, (61, A, A), (8, ALA, A), P3...
(20, ILE, A)	(9, ILE, A)	(62, I, A)	P30044	(62, I, A)	62	62	[(20, ILE, A), 62, (62, I, A), (9, ILE, A), P3...
(21, PRO, A)	(10, PRO, A)	(63, P, A)	P30044	(63, P, A)	63	63	[(21, PRO, A), 63, (63, P, A), (10, PRO, A), P...
(22, ALA, A)	(11, ALA, A)	(64, A, A)	P30044	(64, A, A)	64	64	[(22, ALA, A), 64, (64, A, A), (11, ALA, A), P...

(B)

PDBe	PDB	UniProt	AccessionID	Uni_moD	new_col_Uni	UniProt_50k	Three_Rows_CIF_Num_Uni
(81, PHE, D)	(100, PHE, D)	(82, F, D)	Q4PRK9	(82, F, D)	82	82	[(81, PHE, D), 82, (82, F, D), (100, PHE, D), .
(82, THR, D)	(101, THR, D)	(83, T, D)	Q4PRK9	(83, T, D)	83	83	[(82, THR, D), 83, (83, T, D), (101, THR, D), .
(83, LYS, D)	(102, LYS, D)	(84, K, D)	Q4PRK9	(84, K, D)	84	84	[(83, LYS, D), 84, (84, K, D), (102, LYS, D), .
(84, ALA, D)	(103A, ALA, D)	(85, A, D)	Q4PRK9	(85, A, D)	85	85	[(84, ALA, D), 85, (85, A, D), (103A, ALA, D),.
(85, PRO, D)	(103B, PRO, D)	(86, P, D)	Q4PRK9	(86, P, D)	86	86	[(85, PRO, D), 86, (86, P, D), (103B, PRO, D),.
(86, GLY, D)	(103C, GLY, D)	(87, G, D)	Q4PRK9	(87, G, D)	87	87	[(86, GLY, D), 87, (87, G, D), (103C, GLY, D),.
(87, LYS, D)	(103D, LYS, D)	(88, K, D)	Q4PRK9	(88, K, D)	88	88	[(87, LYS, D), 88, (88, K, D), (103D, LYS, D),.
(88, SER, D)	(103E, SER, D)	(89, S, D)	Q4PRK9	(89, S, D)	89	89	[(88, SER, D), 89, (89, S, D), (103E, SER, D),.
(89, ASP, D)	(105A, ASP, D)	(90, D, D)	Q4PRK9	(90, D, D)	90	90	[(89, ASP, D), 90, (90, D, D), (105A, ASP, D),.
(90, LYS, D)	(105B, LYS, D)	(91, K, D)	Q4PRK9	(91, K, D)	91	91	[(90, LYS, D), 91, (91, K, D), (105B, LYS, D),.
(91, GLU, D)	(106, GLU, D)	(92, E, D)	Q4PRK9	(92, E, D)	92	92	[(91, GLU, D), 92, (92, E, D), (106, GLU, D), .
(92, TRP, D)	(107, TRP, D)	(93, W, D)	Q4PRK9	(93, W, D)	93	93	[(92, TRP, D), 93, (93, W, D), (107, TRP, D), .

(C)

PDBe	PDB	UniProt	AccessionID	Uni_moD	new_col_Uni	UniProt_50k	Three_Rows_CIF_Num_Uni
(1, HIS, A)	(-2, HIS, A)	50000	NaN	(1, HIS, A)	1	50001	[(1, HIS, A), 50001, (1, HIS, A), (-2, HIS, A)...
(2, HIS, A)	(-1, HIS, A)	50000	NaN	(2, HIS, A)	2	50002	[(2, HIS, A), 50002, (2, HIS, A), (-1, HIS, A)...
(3, MET, A)	(1, MET, A)	(1, M, A)	P21856	(1, M, A)	1	1	[(3, MET, A), 1, (1, M, A), (1, MET, A), P21856]
(4, ASP, A)	(2, ASP, A)	(2, D, A)	P21856	(2, D, A)	2	2	[(4, ASP, A), 2, (2, D, A), (2, ASP, A), P21856]
(5, GLU, A)	(3, GLU, A)	(3, E, A)	P21856	(3, E, A)	3	3	[(5, GLU, A), 3, (3, E, A), (3, GLU, A), P21856]
(6, GLU, A)	(4, GLU, A)	(4, E, A)	P21856	(4, E, A)	4	4	[(6, GLU, A), 4, (4, E, A), (4, GLU, A), P21856]
(7, TYR, A)	(5, TYR, A)	(5, Y, A)	P21856	(5, Y, A)	5	5	[(7, TYR, A), 5, (5, Y, A), (5, TYR, A), P21856]
(8, ASP, A)	(6, ASP, A)	(6, D, A)	P21856	(6, D, A)	6	6	[(8, ASP, A), 6, (6, D, A), (6, ASP, A), P21856]
(9, VAL, A)	(7, VAL, A)	(7, V, A)	P21856	(7, V, A)	7	7	[(9, VAL, A), 7, (7, V, A), (7, VAL, A), P21856]
(10, ILE, A)	(8, ILE, A)	(8, I, A)	P21856	(8, I, A)	8	8	[(10, ILE, A), 8, (8, I, A), (8, ILE, A), P21856]
(11, VAL, A)	(9, VAL, A)	(9, V, A)	P21856	(9, V, A)	9	9	[(11, VAL, A), 9, (9, V, A), (9, VAL, A), P21856]

Figure 2. Small fragments of the Pandas dataframes assembled from SIFTS files 2vl3 (A), 2aa3 (B) and 1d5t (C). Entry 2vl3 contains a His tag that is not observed in the coordinates. Chain D of 2aa3

contains insertion codes (column 2) for some residues. Entry 1d5t contains a His tag with negative author residue numbers (column 2). The PDBe column in each image contains data in tuples (SeqResNum, ResName, and ChainID). This field acts as the Pandas dataframe index for the whole table, since it is unique for each amino acid. The PDB column contains tuples (AuthResNum + InsCode, ResName and ChainID). The UniProt column contain tuples of (UniProt ResNum, ResName, ChainID) and if there is no UniProt residue number, it contains the number "50,000". The next column contains the Uniprot AccessionID. Uni_moD is an auxiliary column where it takes the UniProt column data for residues in UniProt, and otherwise the PDBe column data. The new_col_Uni column takes only the ResNum from Uni_moD. The column UniProt_50k adds 50,000 to 1 to N ResNum for the residues where there is no UniProt data and otherwise takes the UniProt ResNum. The column Three_Rows_CIF_Num_Uni combines the data into a Python list [PDBe data, UniProt_50k, Uni_moD, PDB, AccessionID]

Methods

PDBrenum was written in Python with use of Python 3.6, BioPython 1.76 [13], Pandas 0.25.1 [14], and Numpy 1.17 modules within Jupyter-Notebook 6.0.1 [15] and PyCharm 2020.2 (https://www.jetbrains.com/pycharm/) as an integrated development environment on a Ubuntu 20.04 operating system. Figure 1 represents a basic scheme of the PDBrenum workflow. First, PDBrenum downloads the structure files (in PDB or/and mmCIF format) and corresponding SIFTS files (in .xml format). The program downloads files in three attempts; if there is no success in three attempts, the assumption is that there is no such file (sometimes servers might not respond or respond with errors, but it is very unlikely to get three bad responses from the server in a row). PDBrenum then parses the SIFTS file to obtain for each amino acid in each protein chain in the file: the number of each residue in each protein construct numbered from 1 to N (the length of the protein chain) ("SeqResNum" from PDBe:dbResNum in the SIFTS file); the author residue number ("ResNum" from PDB:dbResNum in SIFTS), the insertion codes, if any ("InsCode" from letters in the PDB:dbResNum in SIFTS), author "ChainID" (from PDB:dbChainID in SIFTS), residue name in three letter code ("ResName" from PDB:dbResName), UniProt ID (if any), and UniProt residue number (from UniProt:dbResNum in SIFTS) and amino acid type (in one letter code, from UniProt:dbResName in SIFTS). These are placed in a Pandas dataframe.

The typical Pandas dataframe will look like the ones shown in Figure 2A-C. The dataframe correlates different numbering systems for the amino acids in each protein chain. The unique residue numbering is the 1 to N numbering (SeqResNum) for each chain, where N is the chain length. This is referred to as `label_seq_id` in the mmCIF file coordinate records. The tuple representing (SeqResNum, ResName, and ChainID) is shown in the first column and acts as a Pandas index or key for the table. The second numbering system is that used by the authors, which is represented in the "PDB" column. It consists of tuples (AuthResNum + InsCode, ResName, and ChainID). Insertion codes are letters attached to some residue numbers by authors to create new identifiers for inserted residues in a sequence. They are common in antibody numbering systems [16]. For most amino acids in the PDB, the SIFTS database has a UniProt reference and residue number, which is given in the 3rd column in each dataframe. When there is no UniProt number given in SIFTS (usually for sequence tags), we place the number 50,000 in this column. The resulting numbering system (given in the column labeled "UniProt_50k") that PDBrenum will use as a replacement for the author numbering system is the UniProt number where it is available and 50,000+SeqResNum when there is no UniProt number. This guarantees that there will be no collision between a UniProt residue number, and the numbers assigned to sequence tags and other insertions that are not part of the UniProt numbering system. For any entry or chain that does not have any UniProt annotations, we leave the numbering used by the authors as is, since there is no risk of reusing residue numbers.

auth_mmCIF	mmCIF_copy	PDB	UniProt	AccessionID	Uni_moD	new_col_Uni	UniProt_50k	Three_Rows_CIF_Num_Uni
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(18, THR, A)	(1, THR, A)	(18, THR, A)	(2, T, A)	Q4PRK9	(2, T, A)	2	2	[(1, THR, A), 2, (2, T, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(19, PRO, A)	(2, PRO, A)	(19, PRO, A)	(3, P, A)	Q4PRK9	(3, P, A)	3	3	[(2, PRO, A), 3, (3, P, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(20, LYS, A)	(3, LYS, A)	(20, LYS, A)	(4, K, A)	Q4PRK9	(4, K, A)	4	4	[(3, LYS, A), 4, (4, K, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(21, PRO, A)	(4, PRO, A)	(21, PRO, A)	(5, P, A)	Q4PRK9	(5, P, A)	5	5	[(4, PRO, A), 5, (5, P, A)]
(22, LYS, A)	(5, LYS, A)	(22, LYS, A)	(6, K, A)	Q4PRK9	(6, K, A)	6	6	[(5, LYS, A), 6, (6, K, A)]

Figure 3. Merging pandas dataframes. Pandas merged two dataframe based on two columns "auth_mmCIF" and "PDB" effectively they both represent the same (AuthResNum + InsCode, RenName, ChainID), by matching they give exact number of copies in right order for the replacement.

PDBrenum uncompresses and reads the gzipped coordinate file as a Python dictionary using BioPython. The BioPython dictionary created by the BioPython function MMCIF2DICT forms keys from each mmCIF table and item name as a single string (e.g., "`_atom_site.Cartn_x`"). The value for each key is a Python list (e.g., the x-coordinates for all atoms in an entry). For each table (e.g. "`_atom_site`"), we detect whether the table has author residue numbers, which may be designated as "`auth_seq_id`" or "`auth_seq_num`" and which may be prefixed or suffixed with other identifiers, e.g., "`auth_seq_id_1`", "`beg_auth_seq_id`", or "`pdbx_auth_seq_num`." For each author residue number, we need to create a tuple that with the author residue number uniquely identifies the residue: the insert code ("`ins_code`", "`PDB_ins_code`", or "`pdb_ins_code`", if any), the monomer type ("`auth_comp_id`" with prefixes or suffixes, or

“mon_id”), and the author chain identifier (“auth_asym_id”, “strand_id”, “pdb_strand_id”, or “pdbx_strand_id”). These may occur in any of the following combinations (with or without appended identifiers such as “beg” or “1

The following items in mmCIF files are renumbered:

```
_atom_site_anisotrop.pdbx_auth_seq_id
_atom_site.auth_seq_id
_ndb_struct_na_base_pair.i_auth_seq_id
_ndb_struct_na_base_pair.j_auth_seq_id
_ndb_struct_na_base_pair_step.i_auth_seq_id_1
_ndb_struct_na_base_pair_step.i_auth_seq_id_2
_ndb_struct_na_base_pair_step.j_auth_seq_id_1
_ndb_struct_na_base_pair_step.j_auth_seq_id_2
_pdbx_distant_solvent_atoms.auth_seq_id
_pdbx_refine_tls_group.beg_auth_seq_id
_pdbx_refine_tls_group.end_auth_seq_id
_pdbx_struct_chem_comp_diagnostics.auth_seq_id
_pdbx_struct_conn_angle.ptnr1_auth_seq_id
_pdbx_struct_conn_angle.ptnr2_auth_seq_id
_pdbx_struct_conn_angle.ptnr3_auth_seq_id
_pdbx_struct_mod_residue.auth_seq_id
_pdbx_struct_sheet_hbond.range_1_auth_seq_id
_pdbx_struct_sheet_hbond.range_2_auth_seq_id
_pdbx_struct_special_symmetry.auth_seq_id
_pdbx_unobs_or_zero_occ_atoms.auth_seq_id
_pdbx_unobs_or_zero_occ_residues.auth_seq_id
_pdbx_validate_chiral.auth_seq_id
_pdbx_validate_close_contact.auth_seq_id_1
_pdbx_validate_close_contact.auth_seq_id_2
_pdbx_validate_main_chain_plane.auth_seq_id
_pdbx_validate_peptide_omega.auth_seq_id_1
_pdbx_validate_peptide_omega.auth_seq_id_2
_pdbx_validate_planes.auth_seq_id
_pdbx_validate_polymer_linkage.auth_seq_id_1
_pdbx_validate_polymer_linkage.auth_seq_id_2
_pdbx_validate_rmsd_angle.auth_seq_id_1
_pdbx_validate_rmsd_angle.auth_seq_id_2
_pdbx_validate_rmsd_angle.auth_seq_id_3
_pdbx_validate_rmsd_bond.auth_seq_id_1
_pdbx_validate_rmsd_bond.auth_seq_id_2
_pdbx_validate_symm_contact.auth_seq_id_1
_pdbx_validate_symm_contact.auth_seq_id_2
_pdbx_validate_torsion.auth_seq_id
_struct_conf.beg_auth_seq_id
_struct_conf.end_auth_seq_id
_struct_conn.ptnr1_auth_seq_id
_struct_conn.ptnr2_auth_seq_id
_struct_mon_prot_cis.auth_seq_id
_struct_mon_prot_cis.pdbx_auth_seq_id_2
_struct_ncs_dom_lim.beg_auth_seq_id
_struct_ncs_dom_lim.end_auth_seq_id
_struct_sheet_range.beg_auth_seq_id
_struct_sheet_range.end_auth_seq_id
_struct_site_gen.auth_seq_id
_struct_site.pdbx_auth_seq_id

_pdbx_nonpoly_scheme.auth_seq_num
_pdbx_poly_seq_scheme.auth_seq_num
_struct_ref_seq_dif.pdbx_auth_seq_num
```

PDBrenum forms a Pandas dataframe for each mmCIF table (e.g. _atom_site) with index names equal to the Python tuple consisting of (Auth ResNum + InsCode, ResName, ChainID), and merges it with the same combination (labeled “PDB” in Figure 2) from the SIFTS dataframe, resulting in the table shown in Figure 3. This table is then used to replace the AuthResNum values with the UniProt ResNum or 50,000+SeqResNum values. Non-polymeric molecule types such as small ligands are also renumbered as 50,000+their residue number in the mmCIF file.

For the space-delimited legacy PDB format files, if the line starts with ("ATOM") or ("TER") or ("ANISOU") or ("SIGUIJ") the program gets columns 22:26, 27, 17:20, 21 as residue number (AuthResNum), insertion code (InsCode), residue name (AuthResName), and ChainID respectively. For special lines “REMARK 465” records which list missing residues, columns 20:26, 27, 15:18, 19 are obtained as the residue number, insertion code, residue name, and ChainID. The two data frames are merged and if the residue does not have a UniProt residue number then it gets default_PDB_num (default_PDB_num = 5000 + SeqResNum). In order to prevent possible numbering collisions, PDBrenum calculates available numbers in the range from 1 to 9999 and reassigns them to non-polymeric compounds in reverse order. After PDBrenum makes the replacement dictionary out of two python strings where the keys are AuthResNum (4 Char) + InsCode (1 Char) + ResName (3 Char) + ChainID (1 Char) and the values are UniProtResNum + InsCode + ResName + ChainID, where the values have the same number of characters. Finally, PDBrenum processes each line of the PDB file, replacing keys with values.

The value offset for non-UniProt residues, default_PDB_num (5,000), can be reset (with “--set_default_mmCIF_num” flag) and default_mmCIF_num (50,000) (with “--

set_default_PDB_num" flag) as you wish but we recommend it to be big enough so it will not be the same as any other numbers but not bigger then 9000 for PDB format because it might go over the 4-character limit of 9999).

If there is no SIFTS file or no UniProt data for a particular chain in the SIFTS file, the entry or chain will not be renumbered at all. This occurs frequently for peptides and antibodies that often do not have corresponding UniProt sequences. The same is true for nucleic acids and non-polymeric entities with their own author ChainIDs.

There are some chains in the PDB that are chimeras containing sequence from two or more UniProt entries. In cases where there is no collision of residue numbering, then the sequences are numbered according to the UniProt sequences in the SIFTS file. In cases when there is a collision, the longest sequence is taken as the default. The only exception to this is for a small number of UniProt sequences that are commonly used as crystallization chaperones, and are therefore not the target of main interest. These include UniProt codes GFP_AEQVI, GCN4_YEAST, C562_ECOLX, ENLYS_BPT4, MALE_ECOLI.

Setting up PDBrenum

Prerequisites anaconda should be installed https://docs.anaconda.com/anaconda/install/.
The following will set up a conda environment for running PDBrenum locally:

```
(base) $ git clone https://github.com/Faezov/PDBrenum.git
(base) $ cd PDBrenum
(base) $ conda create -n PDBrenum python=3.6 numpy=1.17 pandas=0.25.1 biopython=1.76
tqdm=4.36.1 ipython=7.8.0 requests=2.25.1 lxml=4.6.2
(base) $ conda activate PDBrenum
```

Running PDBrenum:

```
(PDBrenum) $ python3 PDBrenum.py -h
```

The user can provide PDBids directly as a list of arguments (-rfla --renumber_from_list_of_arguments):

```
(PDBrenum) $ python3 PDBrenum.py -rfla 1d5t 1bxw 2vl3 5e6h -mmCIF
(PDBrenum) $ python3 PDBrenum.py -rfla 1d5t 1bxw 2vl3 5e6h -PDB
(PDBrenum) $ python3 PDBrenum.py -rfla 1d5t 1bxw 2vl3 5e6h -mmCIF_assembly
(PDBrenum) $ python3 PDBrenum.py -rfla 1d5t 1bxw 2vl3 5e6h -PDB_assembly
```

or put PDBids in text file (comma, space or tab delimited) (-rftf --renumber_from_text_file):

```
(PDBrenum) $ python3 PDBrenum.py -rftf input.txt -mmCIF
(PDBrenum) $ python3 PDBrenum.py -rftf input.txt -PDB
(PDBrenum) $ python3 PDBrenum.py -rftf input.txt -mmCIF_assembly
(PDBrenum) $ python3 PDBrenum.py -rftf input.txt -PDB_assembly
```

The user can renumber the entire PDB in a given format (by default in mmCIF if no format was provided):

```
(PDBrenum) $ python3 PDBrenum.py -redb -mmCIF
(PDBrenum) $ python3 PDBrenum.py -redb -PDB
(PDBrenum) $ python3 PDBrenum.py -redb -mmCIF_assembly
(PDBrenum) $ python3 PDBrenum.py -redb -PDB_assembly
```

Note that sometimes on Windows biopython module might be installed incorrectly by conda and it will cause module error in python. To resolve this problem simply run:

```
(PDBrenum) $ pip install biopython==1.76
```

PDBrenum was heavily tested on all PDB structure files in both formats and on all popular operating systems (Linux, Mac and Windows).

PDBrenum uses multiprocessing by default (it will use all available cores) but the user can set a limit to the numer of processors by providing number to -nproc flag.

The user can also change where input output files will go by using these self-explanatory flags (with absolute paths):

```
"-sipm", "--set_default_input_path_to_mmCIF"
"-sipma", "--set_default_input_path_to_mmCIF_assembly"
"-sipp", "--set_default_input_path_to_PDB"
"-sippa", "--set_default_input_path_to_PDB_assembly"
"-sips", "--set_default_input_path_to_SIFTS"
"-sopm", "--set_default_output_path_to_mmCIF"
"-sopma", "--set_default_output_path_to_mmCIF_assembly"
"-sopp", "--set_default_output_path_to_PDB"
"-soppa", "--set_default_output_path_to_PDB_assembly"
```

By default, files go here:

```
default_input_path_to_mmCIF = current_directory + "/mmCIF"
default_input_path_to_mmCIF_assembly = current_directory + "/mmCIF_assembly"
default_input_path_to_PDB = current_directory + "/PDB"
default_input_path_to_PDB_assembly = current_directory + "/PDB_assembly"
default_input_path_to_SIFTS = current_directory + "/SIFTS"
default_output_path_to_mmCIF = current_directory + "/output_mmCIF"
default_output_path_to_mmCIF_assembly = current_directory + "/output_mmCIF_assembly"
default_output_path_to_PDB = current_directory + "/output_PDB"
default_output_path_to_PDB_assembly = current_directory + "/output_PDB_assembly"
```

Also, by default all files gzipped if you want have them unzipped please use `"-offz"` or `"--set_to_off_mode_gzip"`

SP	PDB_id	chain_PDB	chain_auth	comp_uni	human_uni	prot_len	uni_len	chain_len	renum	5k_or_50k
+	2aa3	A	A	Q4PRK9	Q4PRK9_PLAVI	1264	1260	316	315	1
+	2aa3	B	B	Q4PRK9	Q4PRK9_PLAVI	1264	1260	315	315	0
+	2aa3	C	C	Q4PRK9	Q4PRK9_PLAVI	1264	1260	318	315	3
+	2aa3	D	D	Q4PRK9	Q4PRK9_PLAVI	1264	1260	315	315	0

Figure 4. Log file of PDBrenum on PDB entry 2aa3

Result and Discussion

PDBrenum returns a log file and renumbered structure files, shown in Figures 4 and 5 respectively for PDB entry 2aa3. Green arrows pointed to the columns which were changed.

We have created a webserver (http://dunbrack3.fccc.edu/PDBrenum) that will take a list of PDB entry codes or a list of UniProt identifiers (as Accession IDs or SwissProt IDs) and with one click of the mouse, will return a zip file with requested mmCIF or legacy-PDB format files of the asymmetric units and/or biological assemblies (Figure 6). The output files can also be accessed programmatically via direct http links

On December 1, 2020, the PDB had 167,518 structures. When processed with PDBrenum:

1) 3581 structures (2.1%) do not have SIFTS (mostly DNA or RNA-only files)

2) 5,372 structures (3.2%) have SIFTS files but do not have any UniProt data in SIFTS (mostly antibodies)

3) 52,099 structures (31.1%) were not changed because all of the proteins have UniProt numbering (although some sequence tags may be renumbered)

4) 106,484 structures (63.6%) were changed due to differences between the author residue numbering and the corresponding UniProt sequences (in numbering of at least one AA)

The large percentage of files that contain proteins that are not numbered according to a standard for each protein (UniProt) demonstrates the need for the approach enabled by PDBrenum. The problem of inconsistent numbering, insertion codes, negative residue numbers, and other problems have been discussed but not addressed in any rigorous way (e.g. https://proteopedia.org/wiki/index.php/Unusual_sequence_numbering and https://www3.cmbi.umcn.nl/wiki/index.php/Residue_number). Mapping PDB structures to UniProt has been attempted a number of times, including SSMAP [17], Seq2Struct [18], and PDBSProtEC [19], although these servers did not renumber actual coordinate files, instead only providing the mapping of residue numbers from the PDB to UniProt. None of them is currently available.

We feel it is very important that PDBrenum provides renumbered files for both mmCIF and legacy-PDB format. mmCIF is the current standard for PDB files, and provides significantly more information than the original PDB format developed in the 1970s. However, many programs still exist that take only the legacy PDB format, and so it is still worthwhile to provide these files. Finally, we also feel it is important to provide the biological assembly files. More than half of crystal structures in the PDB have annotated assemblies that are different from the asymmetric unit [20]. While RCSB does not provide these files in mmCIF format at this time, they are available from PDBe. Even when the assembly consists of multiple copies of the asymmetric unit, every chain in the assembly files from PDBe has its own unique chainID in the auth_asym_id fields, and so can be processed like any PDB file for further analysis.

Acknowledgments
This work was supported by NIH grant R35 GM122517. We thank Vivek Modi and Mitchell Parker for testing the PDBrenum program.

(A)

```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.pdbx_formal_charge
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM   1     N N   . THR A 1 1   ? 21.071  -3.780  0.959   0.50 33.08 ? 18   THR A N   1
ATOM   2     C CA  . THR A 1 1   ? 20.920  -4.893  -0.028  0.50 33.12 ? 18   THR A CA  1
ATOM   3     C C   . THR A 1 1   ? 20.260  -4.505  -1.390  0.50 32.86 ? 18   THR A C   1
ATOM   4     O O   . THR A 1 1   ? 19.219  -5.071  -1.781  0.50 33.29 ? 18   THR A O   1
ATOM   5     C CB  . THR A 1 1   ? 20.217  -6.140  0.640   0.50 33.37 ? 18   THR A CB  1
ATOM   6     O OG1 . THR A 1 1   ? 20.264  -7.284  -0.235  0.50 34.48 ? 18   THR A OG1 1
ATOM   7     C CG2 . THR A 1 1   ? 18.758  -5.827  1.056   0.50 32.86 ? 18   THR A CG2 1
ATOM   8     N N   . PRO A 1 2   ? 20.845  -3.532  -2.116  0.50 31.64 ? 19   PRO A N   1
ATOM   9     C CA  . PRO A 1 2   ? 21.882  -2.598  -1.716  0.50 30.90 ? 19   PRO A CA  1
ATOM   10    C C   . PRO A 1 2   ? 21.307  -1.477  -0.840  0.50 29.29 ? 19   PRO A C   1
ATOM   11    O O   . PRO A 1 2   ? 20.113  -1.468  -0.510  0.50 29.80 ? 19   PRO A O   1
ATOM   12    C CB  . PRO A 1 2   ? 22.357  -2.013  -3.052  0.50 30.71 ? 19   PRO A CB  1
ATOM   13    C CG  . PRO A 1 2   ? 21.162  -2.030  -3.898  0.50 31.48 ? 19   PRO A CG  1
ATOM   14    C CD  . PRO A 1 2   ? 20.450  -3.321  -3.522  0.50 32.18 ? 19   PRO A CD  1
ATOM   15    N N   . LYS A 1 3   ? 22.165  -0.537  -0.480  1.00 27.74 ? 20   LYS A N   1
ATOM   16    C CA  . LYS A 1 3   ? 21.851  0.470   0.566   1.00 24.76 ? 20   LYS A CA  1
ATOM   17    C C   . LYS A 1 3   ? 20.859  1.474   -0.070  1.00 21.28 ? 20   LYS A C   1
ATOM   18    O O   . LYS A 1 3   ? 21.118  1.943   -1.164  1.00 20.53 ? 20   LYS A O   1
ATOM   19    C CB  . LYS A 1 3   ? 23.173  1.165   0.968   1.00 25.30 ? 20   LYS A CB  1
ATOM   20    C CG  . LYS A 1 3   ? 23.163  2.034   2.252   1.00 25.01 ? 20   LYS A CG  1
ATOM   21    C CD  . LYS A 1 3   ? 23.196  1.149   3.512   1.00 29.51 ? 20   LYS A CD  1
ATOM   22    C CE  . LYS A 1 3   ? 22.595  1.857   4.796   1.00 29.55 ? 20   LYS A CE  1
ATOM   23    N NZ  . LYS A 1 3   ? 22.037  0.855   5.799   1.00 28.73 ? 20   LYS A NZ  1
```

```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.pdbx_formal_charge
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM   1     N N   . THR A 1 1   ? 21.071  -3.780  0.959   0.50 33.08 ? 2    THR A N   1
ATOM   2     C CA  . THR A 1 1   ? 20.920  -4.893  -0.028  0.50 33.12 ? 2    THR A CA  1
ATOM   3     C C   . THR A 1 1   ? 20.260  -4.505  -1.390  0.50 32.86 ? 2    THR A C   1
ATOM   4     O O   . THR A 1 1   ? 19.219  -5.071  -1.781  0.50 33.29 ? 2    THR A O   1
ATOM   5     C CB  . THR A 1 1   ? 20.217  -6.140  0.640   0.50 33.37 ? 2    THR A CB  1
ATOM   6     O OG1 . THR A 1 1   ? 20.264  -7.284  -0.235  0.50 34.48 ? 2    THR A OG1 1
ATOM   7     C CG2 . THR A 1 1   ? 18.758  -5.827  1.056   0.50 32.86 ? 2    THR A CG2 1
ATOM   8     N N   . PRO A 1 2   ? 20.845  -3.532  -2.116  0.50 31.64 ? 3    PRO A N   1
ATOM   9     C CA  . PRO A 1 2   ? 21.882  -2.598  -1.716  0.50 30.90 ? 3    PRO A CA  1
ATOM   10    C C   . PRO A 1 2   ? 21.307  -1.477  -0.840  0.50 29.29 ? 3    PRO A C   1
ATOM   11    O O   . PRO A 1 2   ? 20.113  -1.468  -0.510  0.50 29.80 ? 3    PRO A O   1
ATOM   12    C CB  . PRO A 1 2   ? 22.357  -2.013  -3.052  0.50 30.71 ? 3    PRO A CB  1
ATOM   13    C CG  . PRO A 1 2   ? 21.162  -2.030  -3.898  0.50 31.48 ? 3    PRO A CG  1
ATOM   14    C CD  . PRO A 1 2   ? 20.450  -3.321  -3.522  0.50 32.18 ? 3    PRO A CD  1
ATOM   15    N N   . LYS A 1 3   ? 22.165  -0.537  -0.480  1.00 27.74 ? 4    LYS A N   1
ATOM   16    C CA  . LYS A 1 3   ? 21.851  0.470   0.566   1.00 24.76 ? 4    LYS A CA  1
ATOM   17    C C   . LYS A 1 3   ? 20.859  1.474   -0.070  1.00 21.28 ? 4    LYS A C   1
ATOM   18    O O   . LYS A 1 3   ? 21.118  1.943   -1.164  1.00 20.53 ? 4    LYS A O   1
ATOM   19    C CB  . LYS A 1 3   ? 23.173  1.165   0.968   1.00 25.30 ? 4    LYS A CB  1
ATOM   20    C CG  . LYS A 1 3   ? 23.163  2.034   2.252   1.00 25.01 ? 4    LYS A CG  1
ATOM   21    C CD  . LYS A 1 3   ? 23.196  1.149   3.512   1.00 29.51 ? 4    LYS A CD  1
ATOM   22    C CE  . LYS A 1 3   ? 22.595  1.857   4.796   1.00 29.55 ? 4    LYS A CE  1
ATOM   23    N NZ  . LYS A 1 3   ? 22.037  0.855   5.799   1.00 28.73 ? 4    LYS A NZ  1
```

(B)

```
loop_
_struct_conf.conf_type_id
_struct_conf.id
_struct_conf.pdbx_PDB_helix_id
_struct_conf.beg_label_comp_id
_struct_conf.beg_label_asym_id
_struct_conf.beg_label_seq_id
_struct_conf.pdbx_beg_PDB_ins_code
_struct_conf.end_label_comp_id
_struct_conf.end_label_asym_id
_struct_conf.end_label_seq_id
_struct_conf.pdbx_end_PDB_ins_code
_struct_conf.beg_auth_comp_id
_struct_conf.beg_auth_asym_id
_struct_conf.beg_auth_seq_id
_struct_conf.end_auth_comp_id
_struct_conf.end_auth_asym_id
_struct_conf.end_auth_seq_id
_struct_conf.pdbx_PDB_helix_class
_struct_conf.details
_struct_conf.pdbx_PDB_helix_length
HELX_P HELX_P1  1  GLY A 12  ? LYS A 25  ? GLY A 29  LYS A 43  1 ? 14
HELX_P HELX_P2  2  ASN A 38  ? TYR A 55  B ASN A 57  TYR A 73  1 ? 18
HELX_P HELX_P3  3  SER A 65  ? LYS A 70  ? SER A 84  LYS A 89  5 ? 6
HELX_P HELX_P4  4  ASP A 96  ? CYS A 116 ? ASP A 111 CYS A 131 1 ? 21
HELX_P HELX_P5  5  PRO A 127 ? GLY A 140 ? PRO A 141 GLY A 154 1 ? 14
HELX_P HELX_P6  6  PRO A 142 ? ASN A 144 ? PRO A 156 ASN A 158 5 ? 3
HELX_P HELX_P7  7  GLY A 150 ? ASN A 167 ? GLY A 164 ASN A 181 1 ? 18
HELX_P HELX_P8  8  CYS A 169 ? ARG A 171 ? CYS A 183 ARG A 185 5 ? 3
HELX_P HELX_P9  9  LYS A 189 ? TYR A 191 ? LYS A 203 TYR A 205 5 ? 3
HELX_P HELX_P10 10 LEU A 199 A ASN A 205 ? LEU A 210 ASN A 215 1 ? 7
HELX_P HELX_P11 11 THR A 209 ? ASN A 222 ? THR A 220 ASN A 234 1 ? 14
HELX_P HELX_P12 12 ASN A 222 ? LEU A 231 ? ASN A 234 LEU A 243 1 ? 10
HELX_P HELX_P13 13 TYR A 235 ? LYS A 251 ? TYR A 247 LYS A 263 1 ? 17
HELX_P HELX_P14 14 GLN A 266 ? GLY A 268 ? GLN A 278 GLY A 280 5 ? 3
HELX_P HELX_P15 15 ASN A 294 ? ALA A 313 ? ASN A 308 ALA A 327 1 ? 20
HELX_P HELX_P16 16 GLY B 12  ? LYS B 25  ? GLY B 29  LYS B 43  1 ? 14
HELX_P HELX_P17 17 ASN B 38  ? TYR B 55  B ASN B 57  TYR B 73  1 ? 18
HELX_P HELX_P18 18 SER B 65  ? LYS B 70  ? SER B 84  LYS B 89  5 ? 6
HELX_P HELX_P19 19 ASN B 93  ? ASP B 95  ? ASN B 108 ASP B 110 5 ? 3
HELX_P HELX_P20 20 ASP B 96  ? CYS B 116 ? ASP B 111 CYS B 131 1 ? 21
```

```
loop_
_struct_conf.conf_type_id
_struct_conf.id
_struct_conf.pdbx_PDB_helix_id
_struct_conf.beg_label_comp_id
_struct_conf.beg_label_asym_id
_struct_conf.beg_label_seq_id
_struct_conf.pdbx_beg_PDB_ins_code
_struct_conf.end_label_comp_id
_struct_conf.end_label_asym_id
_struct_conf.end_label_seq_id
_struct_conf.pdbx_end_PDB_ins_code
_struct_conf.beg_auth_comp_id
_struct_conf.beg_auth_asym_id
_struct_conf.beg_auth_seq_id
_struct_conf.end_auth_comp_id
_struct_conf.end_auth_asym_id
_struct_conf.end_auth_seq_id
_struct_conf.pdbx_PDB_helix_class
_struct_conf.details
_struct_conf.pdbx_PDB_helix_length
HELX_P HELX_P1  1  GLY A 12  ? LYS A 25  ? GLY A 13  LYS A 26  1 ? 14
HELX_P HELX_P2  2  ASN A 38  ? TYR A 55  B ASN A 39  TYR A 56  1 ? 18
HELX_P HELX_P3  3  SER A 65  ? LYS A 70  ? SER A 66  LYS A 71  5 ? 6
HELX_P HELX_P4  4  ASP A 96  ? CYS A 116 ? ASP A 97  CYS A 117 1 ? 21
HELX_P HELX_P5  5  PRO A 127 ? GLY A 140 ? PRO A 128 GLY A 141 1 ? 14
HELX_P HELX_P6  6  PRO A 142 ? ASN A 144 ? PRO A 143 ASN A 145 5 ? 3
HELX_P HELX_P7  7  GLY A 150 ? ASN A 167 ? GLY A 151 ASN A 168 1 ? 18
HELX_P HELX_P8  8  CYS A 169 ? ARG A 171 ? CYS A 170 ARG A 172 5 ? 3
HELX_P HELX_P9  9  LYS A 189 ? TYR A 191 ? LYS A 190 TYR A 192 5 ? 3
HELX_P HELX_P10 10 LEU A 199 A ASN A 205 ? LEU A 200 ASN A 206 1 ? 7
HELX_P HELX_P11 11 THR A 209 ? ASN A 222 ? THR A 210 ASN A 223 1 ? 14
HELX_P HELX_P12 12 ASN A 222 ? LEU A 231 ? ASN A 223 LEU A 232 1 ? 10
HELX_P HELX_P13 13 TYR A 235 ? LYS A 251 ? TYR A 236 LYS A 252 1 ? 17
HELX_P HELX_P14 14 GLN A 266 ? GLY A 268 ? GLN A 267 GLY A 269 5 ? 3
HELX_P HELX_P15 15 ASN A 294 ? ALA A 313 ? ASN A 295 ALA A 314 1 ? 20
HELX_P HELX_P16 16 GLY B 12  ? LYS B 25  ? GLY B 13  LYS B 26  1 ? 14
HELX_P HELX_P17 17 ASN B 38  ? TYR B 55  B ASN B 39  TYR B 56  1 ? 18
HELX_P HELX_P18 18 SER B 65  ? LYS B 70  ? SER B 66  LYS B 71  5 ? 6
HELX_P HELX_P19 19 ASN B 93  ? ASP B 95  ? ASN B 94  ASP B 96  5 ? 3
HELX_P HELX_P20 20 ASP B 96  ? CYS B 116 ? ASP B 97  CYS B 117 1 ? 21
```

Figure 5. Screenshot of files 2aa3 before (left) and after (right) PDBrenum run "_atom_site" (coordinate section) (A) and "struct_conf" (B), green arrows pointed to the columns which were changed.

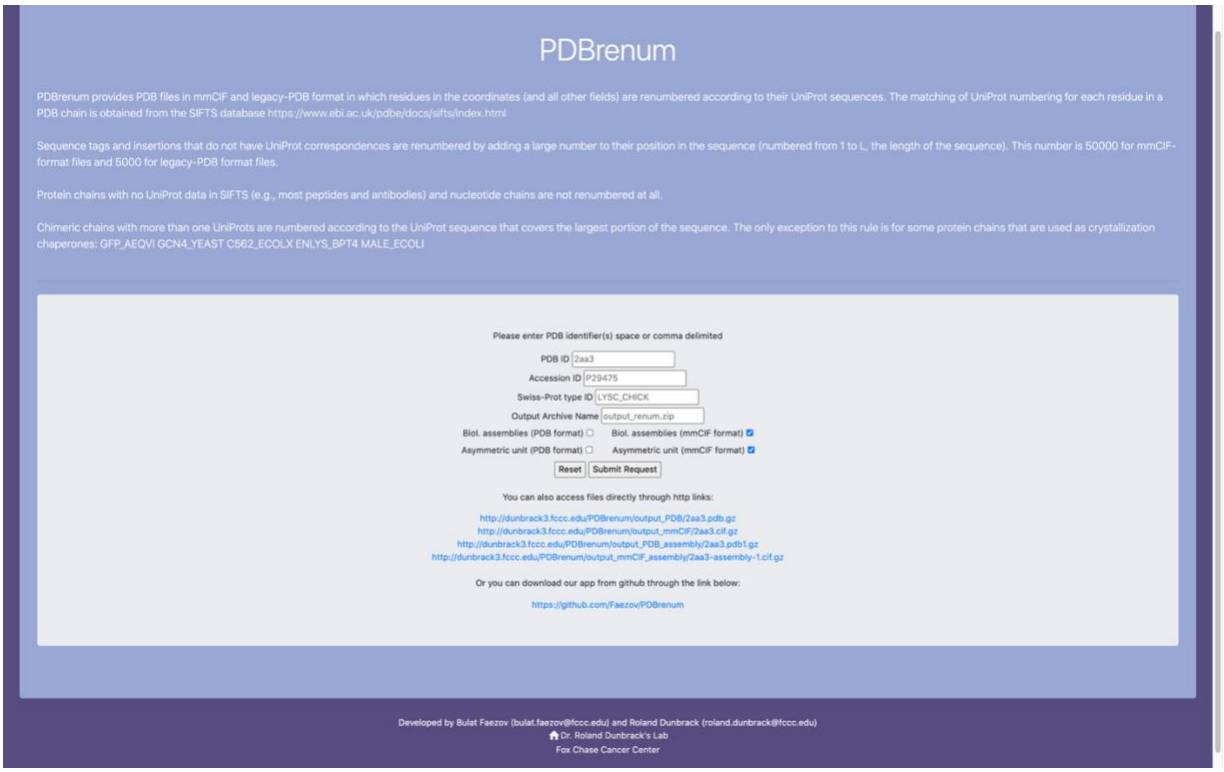

Figure 6. Screenshot of the PDBrenum server.

References

1. Berman H, Henrick K, Nakamura H, Markley JL. The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data. Nucleic Acids Res. 2007;35(suppl_1):D301-D3.
2. Burley SK, Berman HM, Bhikadiya C, Bi C, Chen L, Di Costanzo L, et al. RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. Nucleic Acids Res. 2019;47(D1):D464-D74.
3. Armstrong DR, Berrisford JM, Conroy MJ, Gutmanas A, Anyango S, Choudhary P, et al. PDBe: improved findability of macromolecular structure data in the PDB. Nucleic Acids Res. 2020;48(D1):D335-D43.
4. Kinjo AR, Bekker GJ, Wako H, Endo S, Tsuchiya Y, Sato H, et al. New tools and functions in data-out activities at Protein Data Bank Japan (PDBj). Protein Sci. 2018;27(1):95-102.
5. Sayers EW, Cavanaugh M, Clark K, Ostell J, Pruitt KD, Karsch-Mizrachi I. GenBank. Nucleic Acids Res. 2019;47(D1):D94-D9.
6. UniProt Consortium. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 2019;47(D1):D506-D15.
7. Nemecek C, Metz WA, Wentzler S, Ding FX, Venot C, Souaille C, et al. Design of potent IGF1-R inhibitors related to bis-azaindoles. Chem Biol Drug Des. 2010;76(2):100-6.

8. Wu J, Li W, Craddock BP, Foreman KW, Mulvihill MJ, Ji Qs, et al. Small-molecule inhibition and activation-loop trans-phosphorylation of the IGF1 receptor. The EMBO journal. 2008;27(14):1985-94.
9. Dana JM, Gutmanas A, Tyagi N, Qi G, O'Donovan C, Martin M, et al. SIFTS: updated Structure Integration with Function, Taxonomy and Sequences resource allows 40-fold increase in coverage of structure-based annotations for proteins. Nucleic Acids Res. 2019;47(D1):D482-D9.
10. Young J, Westbrook J, Feng Z, Sala R, Peisach E, Oldfield T, et al. PDBx/mmCIF: the foundation for the wwPDB onedep system. Foundations of Crystallography. 2017;70:C1361.
11. Abola EE, Sussman JL, Prilusky J, Manning NO. [29] Protein data bank archives of three-dimensional macromolecular structures. Methods Enzymol. 1997;277:556-71.
12. Krissinel E, Henrick K. Inference of macromolecular assemblies from crystalline state. J Mol Biol. 2007;372:774-97.
13. Hamelryck T, Manderick B. PDB file parser and structure class implemented in Python. Bioinformatics. 2003;19(17):2308-10.
14. McKinney W, editor Data structures for statistical computing in python. Proceedings of the 9th Python in Science Conference; 2010: Austin, TX.
15. Kluyver T, Ragan-Kelley B, Pérez F, Granger BE, Bussonnier M, Frederic J, et al. Jupyter Notebooks-a publishing format for reproducible computational workflows2016.
16. Abhinandan K, Martin AC. Analysis and improvements to Kabat and structurally correct numbering of antibody variable domains. Mol Immunol. 2008;45(14):3832-9.
17. David FP, Yip YL. SSMap: a new UniProt-PDB mapping resource for the curation of structural-related information in the UniProt/Swiss-Prot Knowledgebase. BMC Bioinformatics. 2008;9(1):1-12.
18. Via A, Zanzoni A, Helmer-Citterich M. Seq2Struct: a resource for establishing sequence-structure links. Bioinformatics. 2005;21(4):551-3.
19. Martin AC. Mapping PDB chains to UniProtKB entries. Bioinformatics. 2005;21(23):4297-301.
20. Xu Q, Dunbrack RL. ProtCID: a data resource for structural information on protein interactions. Nature communications. 2020;11(1):1-16.