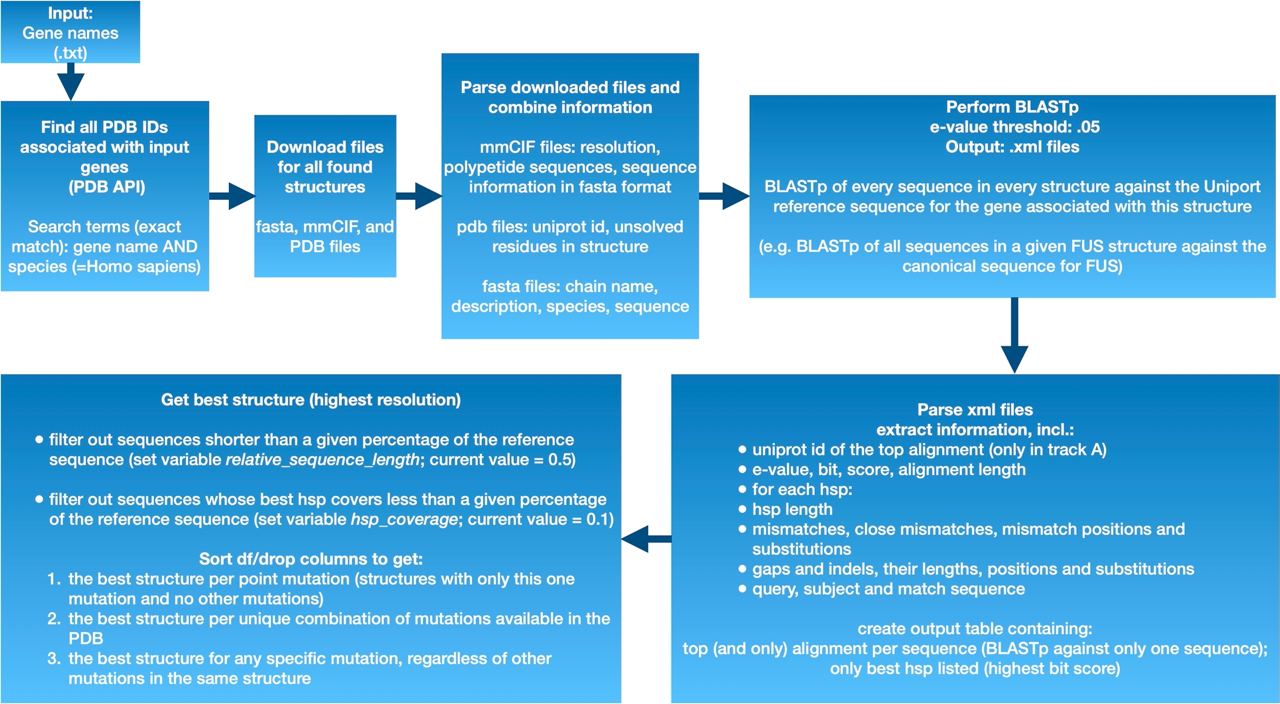
# Overview Pipeline



|  |  |  |  |
| --- | --- | --- | --- |
| Script name | Input | Operations | Output |
| *00\_search\_pdb.py* | * genes in .txt format  (e.g. SOD1 ALS2 FUS) | * Creates a folder called Results in the current working directory where all the output is going to be stored * searches the pdb for all structures associated with each gene name (in Homo Sapiens) | * ***directory: Results/*** * ***00\_search\_overview\_PDBids.csv*** contains all gene names and corresponding PDB IDs if available |
| *01\_download\_files.py* | * ***00\_search\_overview\_PDBids.csv*** | * creates a folder for each gene in the Results directory * downloads mmCIF, pdb, and fasta files for all structures into the respective gene folder | * ***directory per gene: Results/GENENAME/***   In each respective gene folder:   * ***mmCIF, pdb, and fasta files*** stored in format pdbID.cif/pdb/fasta   In the Results folder:   * ***01\_search\_overview\_folders.csv*** lists all the the newly created folders and their contents * ***01\_search\_overview\_n\_structures.csv*** lists number of structures retrieved per gene |
| *02\_parse\_cif\_files.py* | * ***01\_search\_overview\_folders.csv*** | * Loops over all created folders and parses all mmCIf files, extracting:   + Resolution (999 for missing values/NMR structures)   + Polypeptide sequences (corresponds to sequences as shown in PyMOL)   + Fasta sequences | In each respective gene folder:   * ***\_ex.fasta file*** for every structure  fasta file created from the mmCIF file * ***GENENAME\_02\_resolutions.csv*** contains the resolution for each structure associated with this gene * ***GENENAME\_02\_poly\_seq.csv*** contains all polypeptide sequences for all structures associated with this gene   In the Results folder:   * ***02\_all\_resolutions.csv***  contains the resolutions of all parsed structures for all genes * ***02\_all\_poly\_seq.csv***  contains all polypeptide sequences for all structures of all genes |
| *03\_parse\_fasta\_files.py* | * ***01\_search\_overview\_folders.csv*** | * Loops over folders containing fasta and \_ex.fasta files * extracts info from fasta files, incl:   + chain name   + description   + species   + sequence * extracts info from \_ex.fasta files, incl:   + chain name   + description   + uniprot id   + sequence * combines info from the two files | In each respective gene folder:   * ***GENENAME\_05\_fasta\_info.csv***  contains information extracted from all fasta files for this gene * ***GENENAME\_05\_fasta\_ex\_info.csv***  contains information extracted from all \_ex.fasta files for this gene * ***GENNAME\_05\_fasta\_combined\_info.csv*** contains combined information extracted from all fasta and \_ex.fasta files for this gene   In the Results folder:   * ***05\_fasta\_info.csv***  contains information extracted from all fasta files for all genes * ***05\_fasta\_ex\_info.csv***  contains information extracted from all \_ex.fasta files for all genes * ***05\_fasta\_combined\_info.csv***  contains combined information extracted from all fasta and \_ex.fasta files for all genes |
| *04\_blast\_against\_reference.py* | * ***05\_fasta\_combined\_info.csv*** | * Creates a directory called RefSeqs in the Results directory * Loops over csv file  (one row for each unique sequence in all the pdb files for all genes) * downloads the reference sequence for the gene from uniprot into the RefSeqs directory * writes a fasta file for each unique sequence/chain in the df into the RefSeqs directory * performs BLASTp of all sequences against the reference sequence  (e.g. FUS canonical sequence for all sequences in all FUS structures) (output stored in .xml format in RefSeqs) * uses the blast output (xml files) to identify mismatches and add this information to the df | * **directory: Results/Refseqs**   In the RefSeqs folder:   * **reference sequence fasta files** in format GENE\_reference.fasta * **unique sequences fasta files** in format GENE\_pdbID\_Chains.fasta * **BLASTp output files (.xml)** in format GENE\_pdbID\_Chains.xml   In the Results folder:  ***07\_blast\_two\_sequences.csv*** lists all the information in the input file and the corresponding BLASTp results |
| *05\_pdb\_extract\_unsolved\_res.py* | * ***01\_search\_overview\_folders.csv*** * ***06\_blast\_fasta.csv  OR 07\_blast\_two\_sequences.csv*** | * Loops over folders and extracts info on missing residues / residues which have not been solved in the crystal structure from each pdb file * combines information on unsolved residues with info from 06\_blast\_fasta.csv / 08\_blast\_two\_sequences.csv | In the Results folder:   * ***08\_unsolved\_residues\_per\_structure.csv*** lists all unsolved residues in all structures for all genes (one row for each structure) * ***08\_unsolved\_residues\_per\_chain.csv*** lists all unsolved residues in all chains of all structures for all genes (one row for each chain) * **08\_all\_info.csv**  contains all the information from *06\_blast\_fasta.csv* OR *07\_blast\_two\_sequences.csv* plus on unsolved residues extracted from pdb files |
| *06\_best\_structure\_per\_mutation.py* | * ***08\_all\_info.csv*** * **02\_all\_resolutions.csv** | * combines the two dfs (according to PDBid) * filters out sequences shorter than a given percentage of the reference sequence (set variable *relative\_sequence\_length*) * filters out sequences whose best hsp covers less than a given percentage of the reference sequence (set variable *hsp\_coverage*) * sort/filter the df to get:   + the best structure per point mutation (structures with only this one mutation and no other mutations)   + the best structure per unique combination of mutations available in the PDB   + the best structure for any specific mutation, regardless of other mutations in the same structure | In each respective gene folder:   * ***GENENAME\_10\_best\_structure\_per\_point\_mutation.csv*** lists best structure for each point mutation (one mutation per structure) in this gene * ***GENENAME\_10\_best\_structure\_all\_unique\_combinations.csv*** lists best structure for all unique mismatch combinations for this gene * ***GENENAME\_10\_best\_structure\_any\_mutation.csv*** lists best structure for any mismatch in this gene regardless of other mismatches in this structure   In the Results folder:   * ***10\_best\_structure\_per\_point\_mutation.csv*** lists best structure for each point mutation (one mutation per structure) in all genes * ***10\_best\_structure\_all\_unique\_combinations.csv*** lists best structure for all unique mismatch combinations for all genes * ***10\_best\_structure\_any\_mutation.csv*** lists best structure for any mismatch for all genes regardless of other mismatches in this structure |
| *07\_a\_ClinVar\_Annotations\_edirect\_per\_gene\_download\_files.py* |  |  |  |
| *07\_b\_ClinVar\_Annotations\_edirect\_per\_gene\_parse\_files.py* |  |  |  |
| *08\_add\_ClinVar\_Annotations\_to\_best\_structures.py* |  |  |  |