

Cast_variant

- Select a single “canonical” representation of a variant’s effects from the list of all possible effects provided by VEP.
- Identify a variant’s position in uniprot and get an idea of any annotations at that position.
- Identify any PDB structures that are currently available that cover that variant
- Identify any SWISSMODEL that covers that variant
- Provide a simple set of initial structure-based features for the variant such as surface area, secondary structure, and distance to complex members (ligands, other proteins, etc) when available.

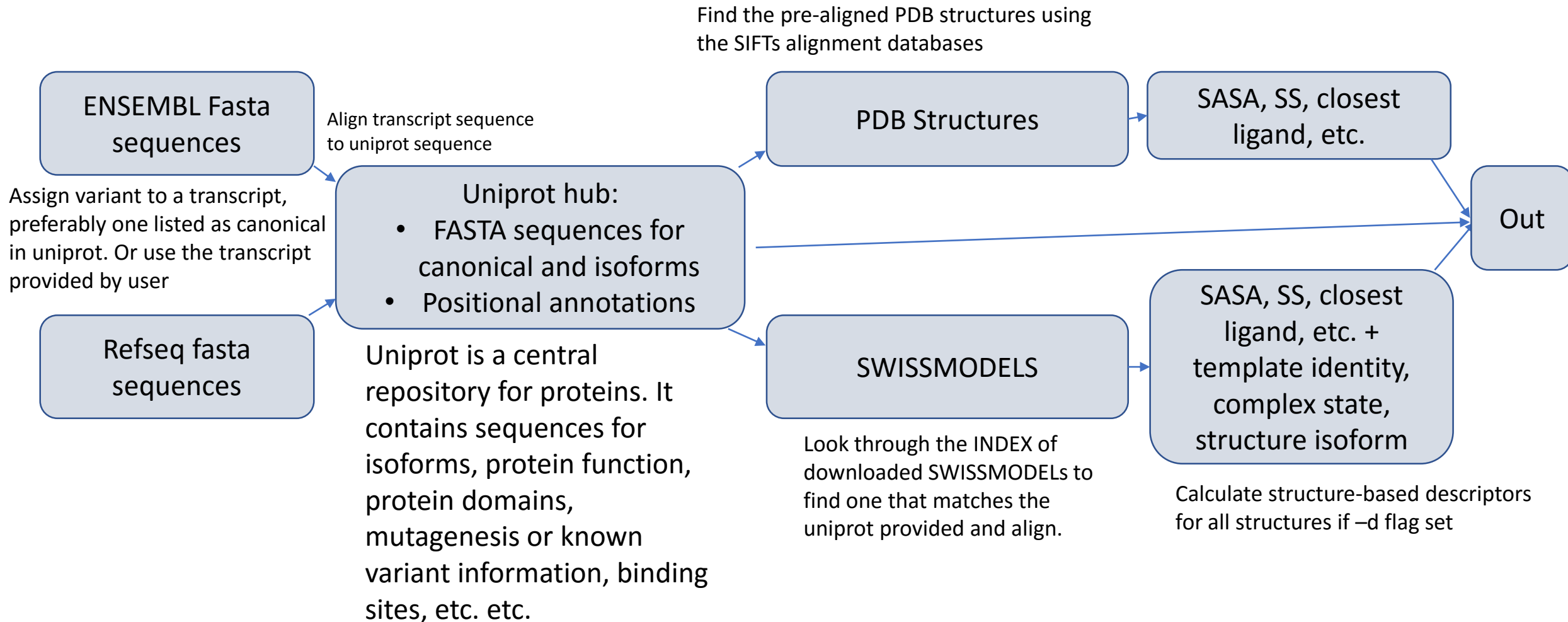
Setting up cast_variant

Dependencies:

python2.7

1. Pandas (pandas)
2. Biopython (bio)
3. Numpy (numpy)
4. xml

Dataset workflow for cast_variants



Required datasets

1) SIFTS: (ftp://ftp.ebi.ac.uk/pub/databases/msd/sifts/split_xml)
<https://www.ebi.ac.uk/pdbe/docs/sifts/>

“Structure Integration with Function, Taxonomy and Sequence (SIFTS) is a project in the [PDBe-KB resource](#) for residue-level mapping between [UniProt](#) and [PDB](#) entries.”

cast_variant:

- SIFTS pre-aligns every PDB structure to the corresponding uniprot entry and is updated regularly. All PDB-uniprot alignments appear in individual xml files that are accessed as needed by cast_variant.
- To quickly identify any SIFTS alignments for a given uniprot, the file **pdb_chain_uniprot.tsv.gz** is downloaded
- This summarizes all SIFTS alignments. For a given uniprot, cast_variant looks in the pickled version of this to see if there are any aligned PDB structures, gets the alignment from the XML file, and reports the aligned position if it covers the variant

Required datasets

2) SWISSMODEL (<https://swissmodel.expasy.org/>)

- This is an automated pipeline service for modeling protein structures.
- It looks at all known protein structures and identifies known structures with an amino-acid sequence that is similar to a given transcript. Then, it uses that known structure to replace all amino acids with those aligned to the given transcript, and uses some limited movement to accommodate these AA's into the existing structure and relieve some energetic clashes. This “comparative modeling” strategy relies on the assumption that similar protein sequences lead to similar folds and that using a known protein with a similar sequence, we can get a pretty good idea of what our protein looks like and only minor adjustments diverge it from the known structure.
- The known structure is called the “template” and can be the same protein in a different species, a similar protein in the same family, or a completely different protein that happens to have a highly similar sequence. Additionally, many modeling pipelines will model the same protein as the template. Often, PDB structures are missing short fragments of a protein for experimental reasons and comparative modeling can take the existing structure and attempt to fill in those residues. In these cases, the template sequence identity is 100%
- The further the sequences differ, the less likely the proteins are to fold into similar structures, therefore, people generally will not use any structure templates that are not at least 30% sequence similar to their transcript of interest.
- SWISSMODEL pipeline iterates through all uniprot isoforms, automatically finds protein structures with a similar sequence, and performs the comparative modeling process. These models are downloaded in bulk to be used.

cast_variant:

- SWISSMODEL database provides an INDEX file that maps all uniprot isoforms to the location of the corresponding model file. This INDEX is processed and used by cast_variant. First, cast_variant looks at the particular isoform being considered, either based on user assignment or during VEP processing. Then, it looks to see if there is a SWISSMODEL particularly for that isoform. If so, it uses that model. If not, it uses all SWISSMODELS for the given uniprot. Typically, this does not matter but in some cases it might so in cases where multiple isoforms exist, one should look at their sequences on uniprot to get an idea of potential differences and also look at the different structures. In most cases, the structures will be extremely similar anyway.

Other models

Currently, `cast_variant` only supports SWISSMODEL structures. This was because `cast_variant` is uniprot-centric and SWISSMODEL is as well.

There are other automated pipelines such as MODBASE that use slightly different approaches to align sequences and select templates as well as performing the minor adjustments during the modeling. No pipeline is consistently “the best” and often the modeled structures look very similar. MODBASE was not used because it is transcript-centric but could easily be added in the future.

Required datasets

3) Ensembl sequences (<https://useast.ensembl.org/info/data/ftp/index.html>)

- Contains all transcript and protein sequences found in the human proteome **as defined by ensembl**
- In some cases, these sequences can differ slightly from Refseq and typically both datasets are queried but only one transcript is used if it is found in both. cast_variant uses ensembl sequences preferentially.
- Transcript sequences are identified by “ENST###” and proteins by “ENSP###” These are almost always identical. Initial set-up processes the main transcript set so finding sequences is faster during runtime.

Required datasets

3) Refseq sequences (<https://www.ncbi.nlm.nih.gov/refseq/about/>)

- Contains all transcript and protein sequences found in the human proteome **as defined by NCBI**
- Transcript sequences are identified by “NM###” and proteins by “NP###” These are almost always identical. Initial set-up processes the main transcript set so finding sequences is faster during runtime.
- Refseq also contains some predicted transcripts that are typically designated with XM### or XP###.

Refseq vs ENSEMBL

- There is no “best” sequence database for the human proteome.
- There are different avenues of annotation and quality control that lead to slight differences between the Refseq and ENSEMBL databases.
- For most cases, only one database is necessary and cast_variant uses ensembl transcript sequences preferentially. However, in some cases, uniprot has selected a Refseq transcript as its canonical sequence. In this case, the Refseq is used.
- Although slight, the differences in these transcript definitions can matter when examining genes and variants. To better understand the Refseq/ENSEMBL headache, there are several publications that compare results from either study:
 - <https://bmcbgenomics.biomedcentral.com/articles/10.1186/s12864-015-1308-8>
 - <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4062061/>

Required datasets

4a) Uniprot sequences

Uniprot is a collection of all known proteins and much of what is known about these proteins. It contains many residue-specific annotations such as where binding sites are, where pathogenic variants are, what is known about the protein's function, binding partners, and much more.

Uniprot collects sequences from both ensembl and Refseq and often defines multiple isoforms from different transcript sequences. Note that when uniprot determines two isoforms of the same gene perform fundamentally different functions, they may split that gene into two individual uniprot entries. This can lead to rare cases where one variant affects multiple uniprot.

cast_variant is uniprot-centric because SIFTS and SWISSMODEL allow easy mapping to structures directly. Additionally, the large collection of residue-specific annotations rely on uniprot's definition of the protein sequence.

The uniprot canonical sequence

- Just as Refseq and ensembl can disagree at the level of the isoform, uniprot has its own “canonical” protein sequence through which it attaches all the residue-level annotations. Therefore, an ‘disulfide bond’ affecting Cys201 in the transcript sequence, may be annotated on uniprot as Cys200 because the first residue of the transcript is a initiator methionine.
- In many cases, the canonical sequence matches a specific ensembl or Refseq sequence. `cast_variant` uses these assignments to select a single variant representation from VEP when possible.
- Because the uniprot canonical sequence may differ from the transcript sequence selected by the user, it is necessary to align the two sequences at the start.
- Note that the SIFTS alignments are relative the uniprot “canonical” sequence. Therefore, it is best to update SIFTS+Uniprot at the same time in case the canonical sequence changes (rare but possible).
- For more information on how uniprot defines its canonical sequence:
 - https://www.uniprot.org/help/canonical_and_isoforms

Required datasets

4b) uniprot canonical-isoform and secondary identifier mappings

- In addition to the multiple isoforms at a given uniprot, each of which may match a different transcript, uniprot continues to change, defining new entries and retiring old ones. In some datasets (VEP, for example), an old or “secondary” uniprot ID will be used. In this case, these libraries can be used to match this old ID to the newest, currently used uniprot ID.

Required datasets

4c) Uniprot XML (uniprot_sprot.xml.gz)

- As mentioned in 4a, uniprot contains many residue-specific annotations that may be useful especially in cases when the variant is not covered by any structures. These annotations are often used as clues to the effect of a given variant and if any are scored True, the user should look at the uniprot entry directly and follow any attached publications which may reveal important information about their variant.
- It is important to note that many annotations are predicted based on well established motifs and other methods and therefore, there may not be a reference but uniprot will indicate it was 'predicted'
- Because the set of annotations is extremely large, initial processing generates a library of annotations for quick lookup during runtime. To parse this large file into the desired data, the processing script requires a significant amount of time but only needs to be performed any time a new uniprot XML is downloaded.

Required application

DSSP (version 2.0.4 provided in repository; newer versions and information <https://swift.cmbi.umcn.nl/gv/dssp/>)

Used to identify secondary structure and solvent accessible surface area of all residues in a given protein structure.

DSSP is used through Biopython which reports surface area as a relative surface area and not the absolute surface area. This is common practice. For more information:

<https://biopython.org/DIST/docs/api/Bio.PDB.DSSP%27-module.html>

The setup process

Follow instructions in `downloading_datasets.txt` in `datasets/`

Update the `config.sys` file to match the paths selected.

Go through the remaining processing steps in `downloading_datasets.txt` that conform the downloaded datasets to that which is expected by `cast_variant` and also runtime speedup processing steps such as **`prepare_uniprot_features.py`** which parses that large uniprot XML to create a library of annotations and **`pickle_sequences.py`** which parses all the downloaded fasta libraries and creates a dictionary for quick lookup during runtime.


The cast_variant workflow

Your variant: rs978862398

Using Variant Effect Predictor to generate input for cast_variant:

<https://useast.ensembl.org/Tools/VEP>

Species:

 Human (Homo sapiens) 

Assembly: GRCh38.p12 (If you are looking for VEP for Human GRCh37, please go to [GRCh37 website](#).)

Name for this job (optional):

Input data:

Either paste data:

17:g.43063897C>T

Run instant VEP for current line ›

Examples: [Ensembl default](#), [VCF](#), [Variant identifiers](#), [HGVS notations](#)

Or upload file:

Choose File No file chosen

Or provide file URL:

Adjust VEP settings before running

cast_variant is best if it can identify the canonical transcript defined in uniprot . This may be a Refseq sequence so allowing VEP to report ENSEMBL and Refseq may be useful in cases where the ENSEMBL transcript is not the canonical in uniprot.

cast_variant attempts to identify the corresponding uniprot entry itself. However, if it is unable to do that, supplying these columns are necessary to identify the corresponding uniprot entry.

Any non-coding variants are immediately filtered anyway

Transcript database to use:

☐ Ensembl/GENCODE transcripts

☐ Ensembl/GENCODE basic transcripts

☐ RefSeq transcripts

☒ Ensembl/GENCODE and RefSeq transcripts

Additional configurations:

Identifiers

Additional identifiers for genes, transcripts and variants

Gene symbol:

☒

CCDS:

☐

Protein:

☒

UniProt:

☒

HGVs:

☐

Variants and frequency data

Co-located variants and frequency data

Additional annotations

Additional transcript, protein and regulatory annotations

Predictions

Variant predictions, e.g. SIFT, PolyPhen

Filtering options

Pre-filter results by frequency or consequence type

Filters

Filter by frequency:

☒ No filtering

☐ Exclude common variants

☐ Advanced filtering

Return results for variants in coding regions only:

☒

Restrict results:

Show all results

NB: Restricting results may exclude biologically important data!

Run

Running VEP

- VEP will use current genome reference and databases containing known gene coordinates, etc to translate your genomic change into a corresponding protein change.
- VEP takes several styles of input, examples are shown by clicking buttons underneath the input box.
- `cast_variant` assumes will not do this prediction as VEP is very good and fast for this step and the online interface is highly recommended. Large variant sets, however, should be broken up into sets of 1000 or so. Those can be submitted in parallel.

VEP Output

Save the TXT file for input to cast_variant, which expects the delimited format of this file.

Results preview

Navigation (per variant)

Page: 1 of 1 | Show: 1 All variants

Filters

Uploaded variant is defined Add

Download

All: VCF VEP TXT

BioMart: Variants Genes

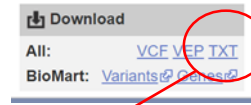
Show/hide columns (9 hidden)

Scroll to see more columns »

Uploaded variant	Location	Allele	Consequence	Impact	Symbol	Gene	Feature type	Feature	Biotype	Exon	cDNA position	CDS position
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000352993	protein_coding	16/22	1822	1703
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000357654	protein_coding	17/23	5361	5129
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000468300	protein_coding	17/22	2011	1817
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000471181	protein_coding	18/24	5424	5192
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000478531	protein_coding	17/18	1919	1817
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000484087	protein_coding	11/12	1442	1442
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000491747	protein_coding	17/23	1916	1817
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000493795	protein_coding	16/22	5220	4988
17:g.43063897C>T	17:43063897-43063897	T	missense_variant	MODERATE	BRCA1	ENSG00000012048	Transcript	ENST00000493919	protein_coding	16/17	1911	1679

VEP will predict all potential consequences based on sequence coordinate databases. Many times there are identical codon positions or amino acid changes as these are slightly different sequences predicted within or across databases. Cast_variant will select **one** representation of each variant based on that which most closely matches uniprot.

Run cast_variant



Saved as

```
./cast_variant.py Example_VEP.txt -d all -v
```

Tells cast_variant you are providing raw VEP output (not needed anymore, should detect automatically, but if not, try using this flag)

This run:

1. Process raw VEP input:
 - For each unique variant, a single representation* is selected and assigned a uniprot ID. This selection is based on using current uniprot ID's and the uniprot-defined canonical sequence whenever possible.
2. Save processed VEP:
 - A file with the extension vep_selected is created. This can be used in future runs in place of the raw VEP TXT file to skip the VEP processing step.
3. Align transcript to uniprot sequence
4. Search for PDB's connected to that uniprot that cover variant position
5. Search for SWISSMODELS connected to that uniprot and isoform (when possible) that cover variant position
6. Calculate accessible surface area and secondary structure for variant in any structure
7. Identify any nearby ligands, DNA, RNA, or other proteins in the structure within a cutoff
8. Identify any uniprot annotations at the variant position and score as True/False

Due to **-d all** flag

*In rare cases, a variant may impact multiple unique uniprot, in these cases, one representation per unique uniprot is kept.

Running VEP

Loaded artifacts: 112 ligands, and 0 protein-protein interfaces

helpers/artifacts.ls defines ligand IDs in PDB structures that are potential crystallization artifacts and are not considered for ligand-distance descriptors

1 unique coding variants loaded

selecting unique variants

YES: ENST

Progression of VEP processing by priority, not important

1 unique variants with uniprot assignment selected

1 Selected vars written to standard varfile example_VEP.vep_selected

Can be used for future runs with same variants to skip previous lines

1 variants processed

loading necessary datasets

loaded 452402 transcript sequences

Loads the pickled transcripts generated from ensembl and refseq fasta downloaded during setup

loaded 20417 uniprot

Loads pickled uniprot sequences downloaded during setup

loaded 46267 sifts uniprot

Loads pickled sifts table of predefined PDB-uniprot alignments downloaded during setup

loaded 17411 swissmodel uniprot

Loads pickled SWISSMODEL lookup table downloaded and generated during setup

Casting variants across 1 transcripts

Casting ENST00000357654 variants

Loading preprocessed uniprot features

Loaded 839479 total residues with uniprot features

For identifying uniprot annotations at variant position from -d all flag from library generated during setup

HTTP Error 404: Not Found

Complete: 1 transcripts successfully aligned

Error explained next slide

HTTP Error 404: Not Found

- In some cases, the variant will fall on a structure that fails to load for a variety of reasons.
- In this case, the variant fell on a very large EM structure (6G2I). Because `cast_variant` retrieves PDBs through URL, and the online database RCSB PDB does not store large structures as PDBs, it is unable to download this structure.
- When a structure fails to load, processing will continue and you can go back and look at those structures manually or see why they failed afterwards.

Examining output

Check for issues first (.skipped and .failures files)

For a variety of reasons, variants may be skipped or something fails when processing them. In these cases, the run continues but they are recorded in one of two ways (in this case, all variants ran successfully so these were not generated)

This is a stop-gain. Currently, VEP only processes single position missense variants. Expand has not been tested, but is designed to treat every residue affected by a single large variant like this one as individual variants

A_skipped_example.skipped

stop_gained	ENST00000272895	1664	L/*	ENSP00000272895	Q86UK0	Q86UK0-1	ABCA12:c.5000T>A	expand not set
-------------	-----------------	------	-----	-----------------	--------	----------	------------------	----------------

A_failure_example.failures

NP_005507.3	P13747	no transcript seq found for NP_005507.3
ENST00000479692	K7ENC6	no unp seq found for K7ENC6
cENST00000335496	Q7Z572	worker: sifts_lookup failed: no sifts entry for Q7Z572

2 Failed during initial sequence alignments because in one case the transcript sequence was not found in the database and in the other the uniprot sequence was not found.

No PDB for Q7Z572, but some output still generated. This is a common failure when a protein has not been crystallized yet. Variant(s) will still appear in output but will not have any PDB entries.

Examining output – columns 1/2

example_VEP.variants

This is the main output for all processed variants. Tab delimited columns:

1. transcript - transcript ID supplied/selected during VEP processing
2. Uniprot – uniprot provided or assigned during VEP processing
3. Isoform – uniprot may be defined with multiple isoforms which may arise from alternative splicing, etc. SWISSMODELS are generated for each (in most cases) isoform and structures may differ slightly.
4. transcript_identity – The identity of the uniprot sequence to the transcript sequence (should be very close to 100%)
5. transcript_position – supplied position in selected transcript
6. uniprot_position – position in the aligned uniprot sequence that can be used to examine annotations on the uniprot website in cases where uniprot position number is different than transcript number.
7. structure_position – position in the structure
8. icode – sometimes experimentors use insertion codes to indicate residues artificially inserted or to keep numbering consistent. These are important when working with raw PDB files, look at external documentation if you have an icode and want to work with the PDB file itself.
9. transcript_aa – amino acid at provided position
10. uniprot_aa – amino acid at aligned uniprot position
11. structure_aa – amino acid at aligned structure sequence
12. ref_aa – reference amino acid
13. alt_aa – variant amino acid

Columns 2/2

4. structure_identity – identity of structure to uniprot sequence, not really useful
5. template_identity - identity of template used to generate SWISSMODEL to uniprot sequence. Very important as it can affect model quality. Generally,, anything below 30% should be ignored as the model quality may be too low.
6. Structure – name of the structure (4 character PDBID or swissmodel file)
7. Chain – identifier within a structure for the particular protein being considered. In multiple protein structures, each protein gets an individual chain. Even if it's a homodimer, each copy of the same protein in the structure gets its own chain. In some cases, you may see a structure show up multiple times for the same variant but with a different chain ID. This is because the variant is covered by multiple chains in the same structure. Many times the structures of those chains are the same, but it is up to the user to determine whether they need to consider every chain a variant hits within a structure or just one.
<https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/dealing-with-coordinates>
8. structure_isoform – As mentioned, each uniprot may have different isoforms and SWISSMODEL attempts to make a structure for each. Therefore, this column only matters for SWISSMODELS and indicates which isoform of the uniprot is being modeled.
https://www.uniprot.org/help/canonical_and_isoforms
9. complex_state – SWISSMODEL uses transcripts from other known structures as templates. Sometimes these structures have multiple proteins in a complex. When a protein is in a complex, it can change conformation. So, this is a warning that the SWISSMODEL template is based on a conformation in a complex which may differ from it by itself.
10. Varcodes – provided or assigned unique variant ID key [chr:position:alt_nucleotide](#)

Descriptor columns: generated with -d flag only

11. SS – secondary structure at position of variant in structure
12. SASA_complex - solvent accessible surface area of variant residue in structure. If structure is a complex of multiple chains, this is the surface area within the complex.
13. SASA_self – solvent accessible surface area of residue in the chain in isolation. Variants at protein-protein interfaces in a complex, for example, will have a very low SASA_complex, but once taken out of the complex it is surface exposed and SASA_self is higher. In non-complex structures, these should be identical.
14. closest_ligand_distance – ligand ID of the closest ligand in the structure (if there is any ligands in the structure)
15. ligands_within_5A - List of all ligand ID's within 5A of variant
16. closest_nucleotide_distance
17. closest_nucleotide_type – DNA vs RNA
18. closest_chain_distance – In structures containing multiple chains (ex complexes), the distance to the closest chain. This will be low for example when the variant is at a protein-protein interface but high when it is on the opposite side of the protein.
19. chains_within_5A
20. cleaved - uniprot annotated as a residue cleaved during protein processing (T/F)
21. sorting - uniprot annotation that residue is part of a sorting motif (T/F)
22. Membrane – uniprot annotation that residue is within the membrane (T/F)
23. binding region – uniprot annotation that residue is within a binding region (binding regions are larger, more general, and/or less precise than binding sites) (T/F)
24. Motif – uniprot annotation that variant is part of a defined motif (T/F)
25. binding site – uniprot annotation that variant participates in binding site (generally higher resolution info than binding region) (T/F)
26. Ptm – uniprot annotation that variant is on residue that undergoes post-translational modification (phosphorylation, glycosylation, etc) (T/F)
27. Covalent – uniprot annotation that variant residue participates in a covalent bond (disulfide bonds are the most common) (T/F)

Overall Output Format

For each variant, there are three potential different kinds of rows. To keep columns consistent, there are times that a column is not applicable to the given row type.

For any column that does not have a value, to keep column types constant, -999 is used for float columns and ? is used for text columns.

Row types:

1. Uniprot – Output for every variant whether or not it has protein coverage. Useful for identifying uniprot position and annotations when no structure is found. Structure-dependent columns (SASA, ligands_within_5A, etc) are not applicable so placeholders are used.
2. PDB – Output for every chain in every RCSB PDB that the variant is covered by
3. SWISSMODEL – output for every SWISSMODEL that variant is covered by

Example output: First data line

This is the first type of output row, the uniprot output

Sequence identifiers				Sequence Positions			Sequence AA's					
ENST00000357654 P38398 P38398-1 1.0				1710	1710	1710	G G -					
ref_aa	alt_aa	Identities are all 100% because there are no structures or templates for uniprot lines		Indicates it's a Uniprot data row so there is no structure name or chain		No complex state since not structure		Unique variant ID chr:pos:alt				
		G	E	100.0	100.0	Uniprot	-	P38398-1	Uniprot	17:43063897-43063897:T		
Placeholders used for all structure-dependent features since this is the uniprot data row and not a structure												
?	-999 False	-999 False	-999 False	?	-999	?	-999	?	False	False	False	False
This variant position does not have any relevant annotations on uniprot (all False). When any of these are True, it is useful to go directly to uniprot and examine the annotations assigned to this uniprot position to see if any information can be quickly gathered.												

Second line – PDB data row

[illegible]

DSSP codes

Code	Structure
H	Alpha helix (4-12)
B	Isolated beta-bridge residue
E	Strand
G	3-10 helix
I	Pi helix
T	Turn
S	Bend
-	None

Last line – SWISSMODEL line

ENST00000357654	P38398	P38398-1	1.0	1710	1710	1710		G	G	G
Template is 100% similar to modeled sequence. In many cases, SWISSMODEL will use a PDB of the same protein to model the protein when there already is a PDB. This can be useful to fill in missing residues absent from the PDB but is generally very similar to the PDB and can be ignored.										
G	E	100.0	100.00	/dors/capra_lab/data/swissmodel/2018-07-23/SWISS-						
MODEL_Repository/P3/83/98/swissmodel/1646_1859_4y18.1.A_5b4ed068c4494f26dcaa3793.pdb										A
Model is specific for -1 isomer			Template is in monomer state			In model, residue is slightly more buried than the PDB. Model may be inaccurate or PDB may be missing portions that would otherwise bury the residue more				
P38398-1	monomer		17:43063897-43063897:T	T	0.25	0.25	-999.0	?		-999.0
?	-999.0	?	False	False	False	False	False	False	False	

Model has no ligands, nucleotides, or other chains, so placeholders used

Other output files

example_VEP.alignments = contains entire sequence alignments to examine any gaps, etc. that may be important to future analysis.

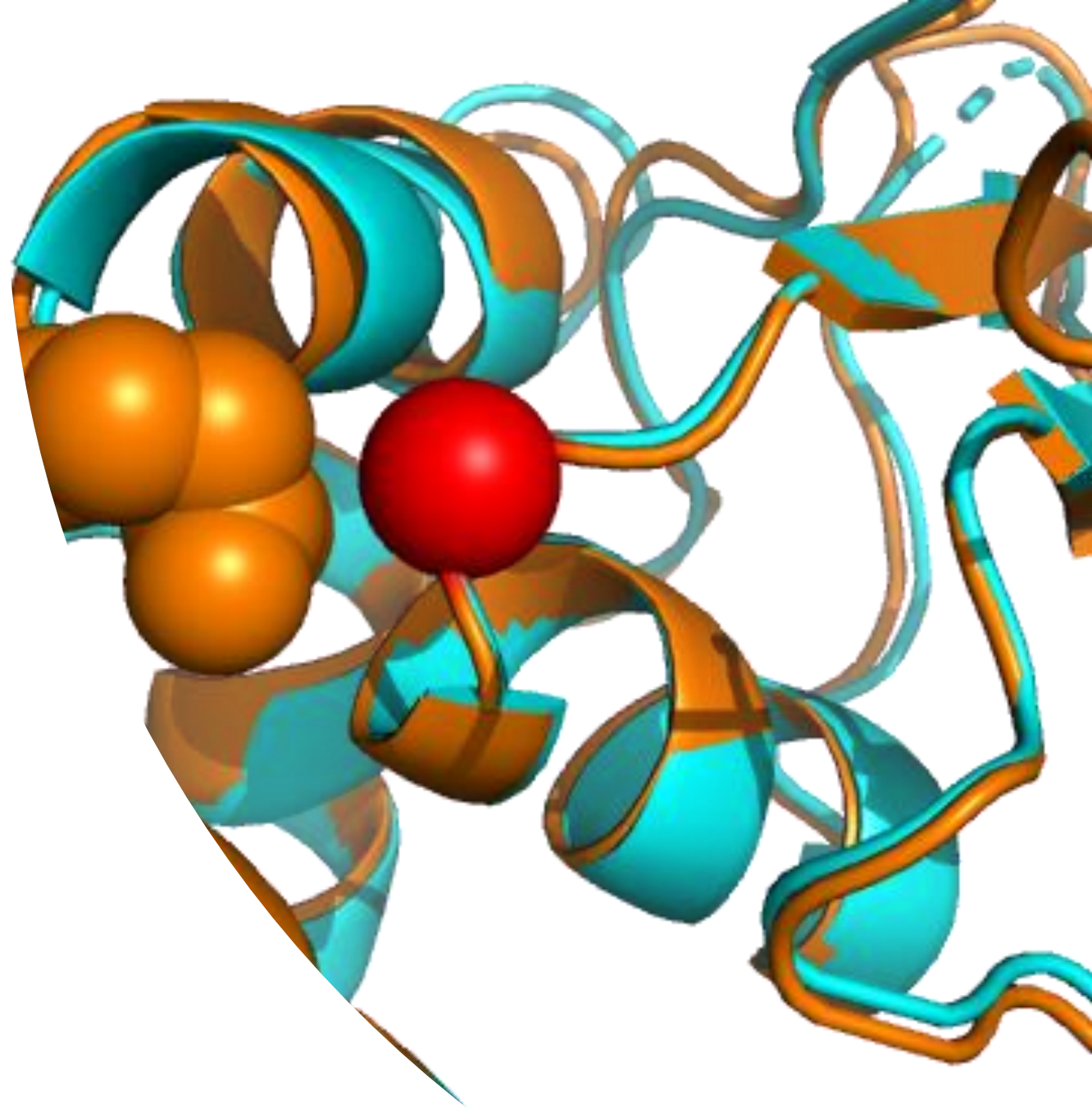
- This file can be very large and you may not be interested in it, so use **-a** to suppress creation of this file

example_VEP.completed = simple file that logs every variant that has been run. Used automatically to continue where program left off if it does not complete for some reason

- suppress the continuation to start entire variant list over with **-c**

Model vs PDB

- Often the PDB is preferred over a model since it has a higher resolution and confident.
- In some cases, especially when identity is extremely high, a model can be preferred.
- In the example, the model had slightly lower surface area than the PDB.
- Examining the position of the variant (red sphere) within the PDB (cyan) and model (orange) reveals that the PDB file does not contain side chain residues aside from CB for Q1756. However, when the model fills in these side chain residues (orange spheres), the surface area of the variant slightly decreases. Therefore, in this case, the calculated surface area of the model is potentially more accurate because it is more complete than the PDB.



Using custom local models

When you want to cast to local models not found in SWISSMODEL, use the `-l` flag with a file containing a list of structure files.

Ex:

```
./cast_variant.py myvars.tab -l mystructs.tab
```

Format of the list of structure files (must be tab delimited):

```
TRANSCRIPT_ID1  MODEL_FILE1.pdb(full path)
TRANSCRIPT_ID1  MODEL_FILE2.pdb(full path)
TRANSCRIPT_ID2  MODEL_FILE3.pdb(full path)
```

This list can contain any number of transcripts and model files. Variants are provided exactly the same, and the transcript column is used to match the model files. This will run variants normally but after casting to SWISSMODELS it will cast to any additional local custom models that you listed with a matching transcript.

Adding custom sequences

Potentially in cases where models are used, you may want to use a custom sequence the model is based on. Therefore, you can add custom sequences to the transcript or uniprot datasets.

This must be done after running `pickle_sequences.py`

In datasets, run:

```
./add_custom_sequences.py your_seqs.fasta destination.pickle
```

Your sequence file can contain any number of fasta's as long as they have a header line in the format:

> Custom identifier

Note: If that identifier is already found in whatever destination you supplied, it will, by default overwrite that entry and tell you. If you don't want to add sequences if the ID is already there, add a 1 as the third argument of the command line.

Untested features

1. Expand variants (-e)

- The major consequence encountered will be missense variants. However, there may be instances in which a large portion of the protein is lost due to early stops, inframe deletions, and frameshifts. In these cases, it can be assumed in most cases that every residue downstream of the mutation is affected, thereby generating a list of individual variant positions. Setting this flag will accept these large variants and expand them to cover all downstream residues. However, it hasn't been tested and may not yet work.

2. Parallel processing (-n #)

- Uses python multiple instance pool to process individual uniprot entries in parallel. This hasn't been tested but should work, maybe, who knows.

A final note on uniprots

It's not necessary to have a uniprot in the uniprot column of the variant file. Things like -, ?, None, etc. are also automatically flagged as missing.

In the case of missing uniprots, `cast_variant` will attempt to assign a uniprot based on the transcript ID.

Future directions

Add modbase

Add more structure-specific features

Add bioassembly integration to filter out protein-protein interaction crystallization artifacts in the same way ligands are filtered.