From sequencing to shortlist

A computational framework for prioritizing protein-coding variants for functional validation

Daniel Angelo R. Mirador

12/10/25

Problem

- Suppose that we have a list of SNPs per gene. How can we shortlist promising protein-coding variants for downstream analysis?
  - Begin with a list of SNPs per gene and identify which ones alter the protein sequence.
  - Identify degree of conservation on the nucleotide and amino acid-level.
  - Annotate variants against UniProt features, topology, and naturally occurring pathogenic sites.
  - Map each amino-acid substitution onto known domains, transmembrane helices, regulatory regions, and functional hotspots.
  - Quantify physicochemical changes (charge, polarity, volume, hydropathy, helix propensity, Grantham distance) to estimate structural or gating impact.
- **Prioritize variants whose predicted effects warrant downstream electrophysiology or *in vivo* functional studies.**

Intro: Python

- We are dealing with dozens to hundreds of sodium channel variants variants, each needing multiple layers of information: coding consequence, nucleotide and amino-acid conservation, structural context (UniProt), physicochemical change, and pathogenicity predictions (ClinVar/VEP).
  - **Doing this manually (per rsID, by hand, on multiple websites) is slow, error-prone, and essentially impossible to reproduce consistently across variants and over time.**
- Python acts as the “glue” that connects all these resources in a standardized way.
- The goal is to provide a transparent, data-driven, and reproducible filter that narrows the variant list to the most mechanistically promising candidates for *in vitro* and *in vivo* follow-up.

Why Python

- Handles API-heavy workflows cleanly (Ensembl, NCBI, UniProt).
- Strong support for sequence processing (Biopython) and large genomic files (pyBigWig).
- Ideal for building automated, reproducible pipelines across many variants.
- Integrates multiple tools (FASTA parsing, conservation, pathogenicity, UniProt mapping) in one environment.
- Ensures consistency and transparency when shortlisting Brugada-related variants.

Why not R

- R is excellent for statistics and visualization, but less suited for:
  - frequent REST API calls,
  - manipulating FASTA/sequence data,
  - accessing bigWig conservation tracks,
  - orchestrating multi-step computational pipelines.
  - Python’s libraries are more mature for bioinformatics automation, making it the better choice for preprocessing and variant annotation.
- *(Also, I am a devoted hater of R…)*

Use of APIs

- API stands for Application Programming Interface.
- An API as a bridge that allows programs to request specific information directly from large databases **without using a website manually.**
- Our pipeline relies on public biological databases that expose their data through REST APIs rather than downloadable static files (which can get heavy).

Use of APIs

- Python allows us to automatically query:
  - Ensembl (coding consequences, transcripts, VEP predictions)
  - NCBI dbSNP / ClinVar (clinical significance)
  - UniProt (domain architecture, topology, PTMs, natural variants)
  - UCSC bigWig tracks (nucleotide conservation)
  - APIs ensure we always retrieve the most current annotations, avoid manual lookup, and standardize processing across all variants.
  - This enables a reproducible, updatable, and scalable approach to shortlisting Brugada-associated variants.

Python use

- Python is a free, widely used programming language in scientific research. Installing it on macOS or Windows provides access to all scripts used in our Brugada variant-analysis pipeline.
- Installation typically involves installing:
  - Python 3 from python.org
  - A package manager like `pip`
  - Required libraries (e.g., requests, biopython, pyBigWig) using simple commands such as:
    ```
    pip3 install biopython pyBigWig requests
    ```

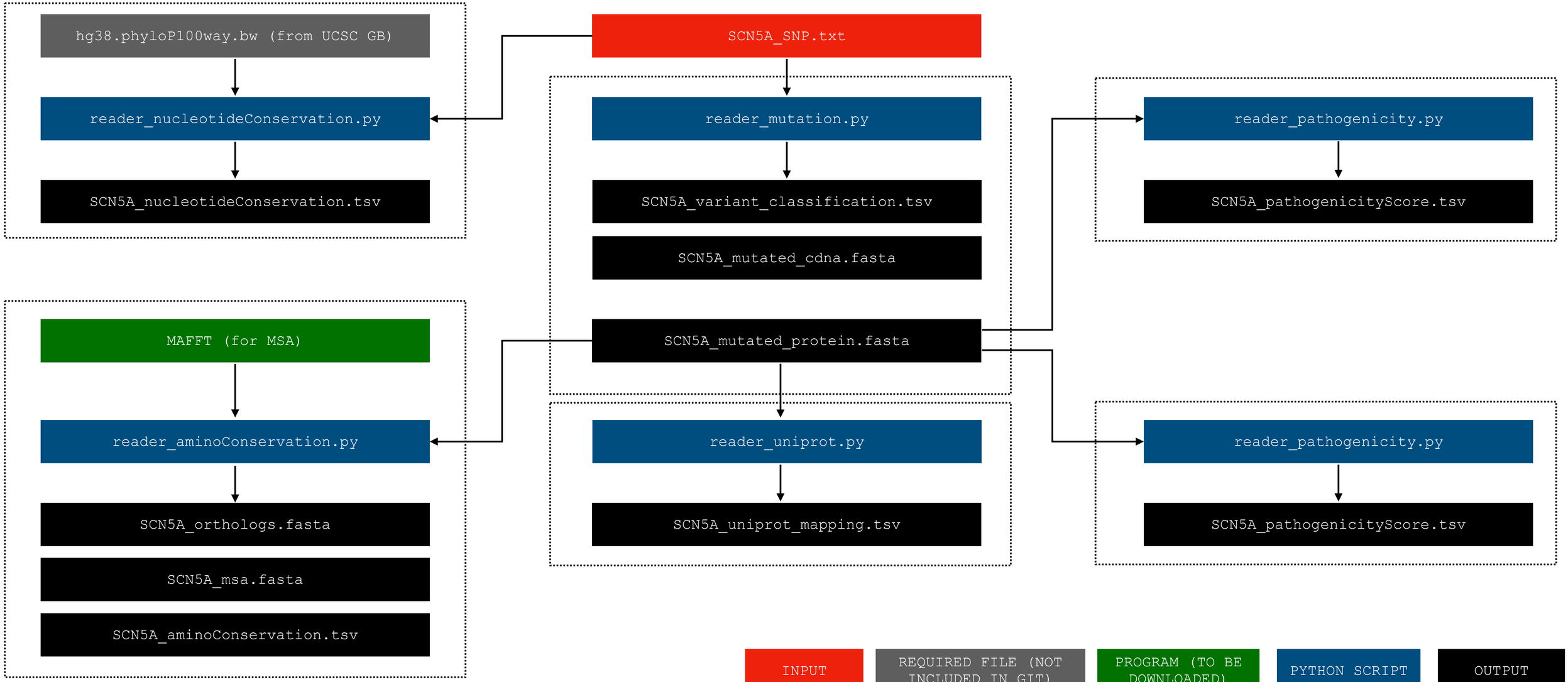
Terminal use

- The terminal (macOS) or Command Prompt/PowerShell (Windows) is simply a place to type commands that run programs—similar to pressing “Run,” but for scripts.
- Running any script looks like:
  `python3 scriptname.py -i inputfile -o outputfile`
  The terminal processes the command, executes the Python code, and produces the requested output files.
- This allows complex analyses (annotation, conservation scoring, UniProt mapping) to be executed reliably and repeatedly, without manual website-based steps.

Terminal use

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
(.venv) (base) darmirador@192 SNP Classification and FASTA % git push origin main
To https://github.com/darmirador/Brugada-Project-Mutation-Analysis.git
   4b4f502..0258b29  main -> main
(.venv) (base) darmirador@192 SNP Classification and FASTA % git add .
(.venv) (base) darmirador@192 SNP Classification and FASTA % git commit -m "edited comments"
[main 1d06580] edited comments
 2 files changed, 3 insertions(+), 3 deletions(-)
(.venv) (base) darmirador@192 SNP Classification and FASTA % git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 403 bytes | 403.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/darmirador/Brugada-Project-Mutation-Analysis.git
   0258b29..1d06580  main -> main
(.venv) (base) darmirador@192 SNP Classification and FASTA %
```

Pipeline (for *SCN5A*, for example)



Input file

- One rsID per line (everything after the semicolon will be ignored)
- Duplicates are acceptable
- Empty lines are also acceptable
- Recommended file name: `[GENE]_SNP.txt` (i.e., `SCN5A_SNP.txt`)

SCN10A_SNP.txt

```
1   rs78425180;95403
2   rs6790627;380868
3   rs4076737;669291
4   rs6795970;259996
5   rs77787358;1264403
6   rs7374804;380869
7   rs7618620;674523
8   rs72866265;674522
9   rs12494065;674521
10  rs7630989;380956
11  rs7617919;259992
12  rs6798015;669276
13  rs7641844;260000
14  rs74717885;381545
15  rs73826327;674520
16  rs62244081;670898
```

SCN5A_SNP.txt

```
1   rs11414422
2   rs41315487;345077
3   rs397763929
4   rs4073796;345087
5   rs7429945;345108
6   rs1805126;48307
7   rs12715317;1291282
8   rs9825620;1230313
9   rs6804918;671861
10  rs41310771;681009
11  rs1805124;48289
12  rs7428779;257437
13  rs41312433;48280
14  rs6770569;671842
15  rs9873213;671841
16  rs6797133
17  rs6781731;671840
18  rs7431305;671839
19  rs11719543;671975
20
21  rs41315485;345075
22  rs4073796;345087
23  rs7429945;345108
24  rs1805126;48307
25  rs6804918;671861
26  rs1805124;48289
```

Using reader_mutation.py

- Description: Classifies SNPs into protein-coding and nonprotein-coding variants
- Terminal:

```
python3 reader_mutation.py -g SCN5A -i SCN5A_SNP.txt -c
SCN5A_mutated_cdna.fasta -p SCN5A_mutated_protein.fasta
```

- Where:
  - `-g` represents the approved gene symbol of the gene (i.e., SCN5A gene for NaV1.5)
  - `-i` represents the input filename
  - `-c` represents the output fasta filename containing the mutated cDNA sequences of *all* entries that have changes in the cDNA sequence (last fasta entry is the wild-type sequence; includes protein-coding and synonymous mutations)
  - `-p` represents the output fasta filename containing the mutated protein (last fasta entry is the wild-type sequence; includes protein-coding mutations)

Using reader_mutation.py

- Outputs one more file, `SCN5A_variant_classification.tsv`, which summarizes the nature of each SNP
- Logic:
  - Takes a list of rsIDs for one human gene (e.g., SCN5A) and looks up each variant in Ensembl using official REST APIs.
  - Automatically identifies the canonical protein-coding transcript and retrieves its reference cDNA, coding sequence (CDS), and protein sequence.
  - For each rsID, determines whether it falls in the coding region and classifies its consequence (e.g., missense, nonsense, synonymous, or noncoding).
  - When a coding change is present, “edits” the reference CDS/cDNA and protein in silico to generate the corresponding mutant sequences.
  - Outputs FASTA files containing all mutated cDNA and protein sequences plus the wild-type reference, alongside a TSV summary that records, for each variant, **its consequence class and whether cDNA and/or protein were altered.**

Using reader_mutation.py

SCN5A_mutated_cdna.fasta

```
>SCN5A|rs1805126|NA|mut_cDNA|ENST00000423572
AGAGCCGCGGAGCCGAGACGGCGGCGGCGCCCGTAGGATGCAGGGATCGCTCCCCCGGGGCCGCTGAGCCTGCGCCCAGTGCCCCGAGCCCCGCGCCGAGCCGAGTCCGCGCCAAGCAGCAGC
>SCN5A|rs1805124|H558R|mut_cDNA|ENST00000423572
AGAGCCGCGGAGCCGAGACGGCGGCGGCGCCCGTAGGATGCAGGGATCGCTCCCCCGGGGCCGCTGAGCCTGCGCCCAGTGCCCCGAGCCCCGCGCCGAGCCGAGTCCGCGCCAAGCAGCAGC
>SCN5A|WT|ENST00000423572|wildtype_cDNA
AGAGCCGCGGAGCCGAGACGGCGGCGGCGCCCGTAGGATGCAGGGATCGCTCCCCCGGGGCCGCTGAGCCTGCGCCCAGTGCCCCGAGCCCCGCGCCGAGCCGAGTCCGCGCCAAGCAGCAGC
```

SCN5A_mutated_protein.fasta

```
>SCN5A|rs1805124|H558R|mut_protein|ENST00000423572
MANFLLPRGTSSFRRFTRESLAAIEKRMAEKQARGSTTLQESREGLPEEEAPRPQLDLQASKKLPDLYGNPPQELIGEPLEDLDPFYSTQKTFIVLNKGKTIFRFSATNALYVLSPFHPIRRA
>SCN5A|WT|ENST00000423572|wildtype_protein
MANFLLPRGTSSFRRFTRESLAAIEKRMAEKQARGSTTLQESREGLPEEEAPRPQLDLQASKKLPDLYGNPPQELIGEPLEDLDPFYSTQKTFIVLNKGKTIFRFSATNALYVLSPFHPIRRA
```

SCN5A_variant_classification.tsv

variant	gene	category	amino_acid_change	note
rs6799868	SCN5A	intron_variant	NA	does not map to CDS
rs6770569	SCN5A	intron_variant	NA	does not map to CDS
rs9873213	SCN5A	intron_variant	NA	does not map to CDS
rs6797133	SCN5A	intron_variant	NA	does not map to CDS
rs6781731	SCN5A	intron_variant	NA	does not map to CDS
rs7431305	SCN5A	intron_variant	NA	does not map to CDS
rs41315485	SCN5A	3_prime_UTR_variant	NA	does not map to CDS
rs11414422	SCN5A	3_prime_UTR_variant	NA	does not map to CDS
rs41315487	SCN5A	3_prime_UTR_variant	NA	does not map to CDS

Using reader_nucleotideConservation.py

- Description: Computes nucleotide conservation scores (phyloP100way) for each SNP by mapping rsIDs to GRCh38 genomic coordinates and querying a local phyloP bigWig file.
- Terminal:

```
python3 reader_nucleotideConservation.py -i SCN5A_SNP.txt -b hg38.phyloP100way.bw -o SCN5A_phyloP.tsv
```

- Where:
  - `-i` represents the input SNP list filename (rsID at the start of each line; any text after ; is ignored).
  - `-b` represents the path to the local phyloP bigWig file (e.g., `hg38.phyloP100way.bw`) downloaded from UCSC and stored in the working directory; this file is large and intentionally excluded from version control.
  - `-o` represents the output TSV filename containing, for each rsID, its GRCh38 genomic location (chromosome:position) and the corresponding phyloP100way conservation score.

Using reader_nucleotideConservation.py

- Logic:
  - Reads a list of rsIDs and queries Ensembl's REST API to resolve each variant's exact GRCh38 genomic position.
  - For every rsID, extracts its chromosome and base-pair coordinate in UCSC-style format (e.g., chr3:1234567).
  - Opens a local phyloP100way bigWig file and retrieves the conservation score at that precise genomic position.
  - Handles missing mappings, unmapped variants, or regions with no phyloP data by reporting NA while continuing the analysis.
  - Writes a TSV output file containing the rsID, its mapped genomic coordinate, and its phyloP conservation score, **enabling downstream integration of nucleotide-level evolutionary constraint into variant prioritization.**

Using reader_nucleotideConservation.py

- Recall:
  - (+) values: more conserved than usual
  - (-) values: less conserved than usual
- Link to bw file:
  - https://hgdownload.cse.ucsc.edu/goldenpath/hg38/phyloP100way/hg38.phyloP100way.bw

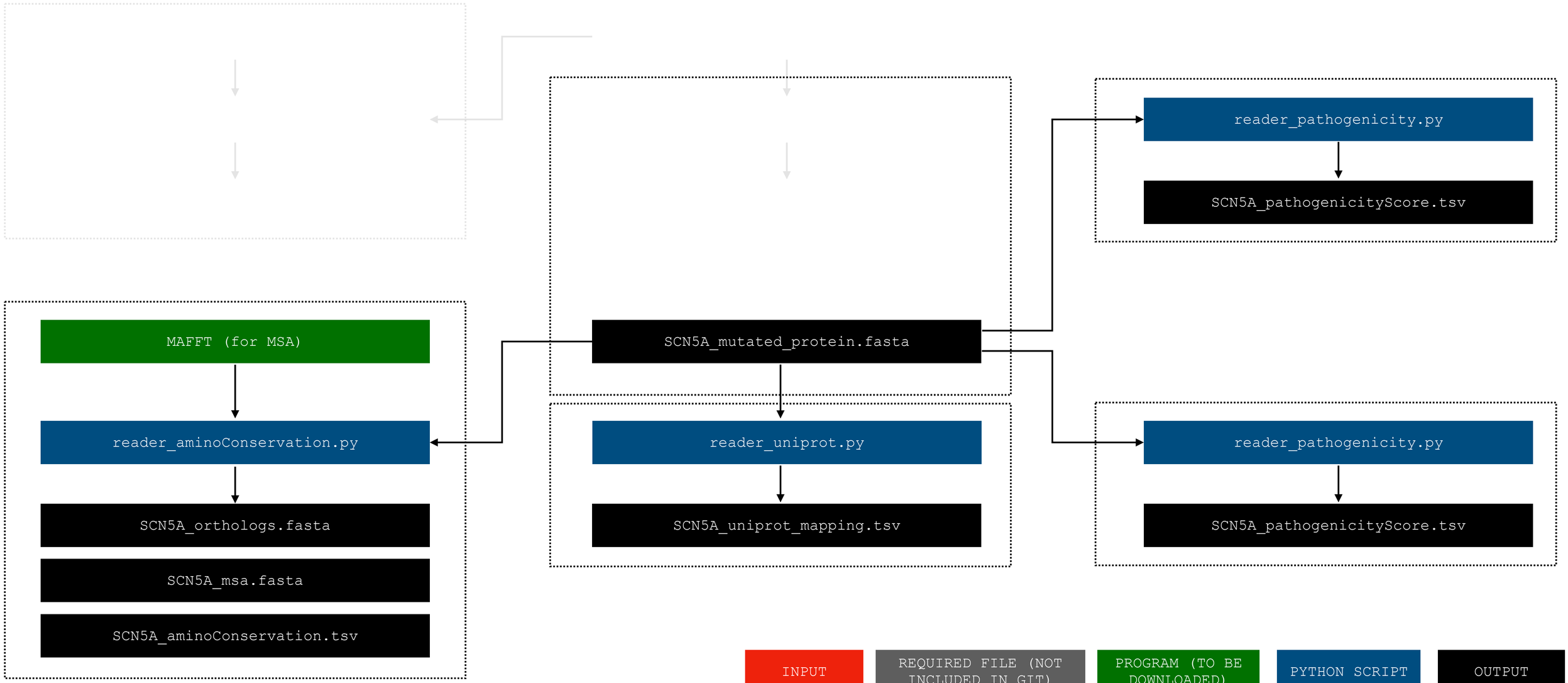
SCN5A_nucleotideConservation.tsv

	rsID	genomic_location	phyloP100way
2	rs11414422	chr3:38548868	0.061000
3	rs41315487	chr3:38548868	0.061000
4	rs397763929	chr3:38548868	0.061000
5	rs4073796	chr3:38549358	-0.240000
6	rs7429945	chr3:38550198	-4.575000
7	rs1805126	chr3:38550915	-2.119000
8	rs12715317	chr3:38555570	-1.495000
9	rs9825620	chr3:38555854	-0.681000
10	rs6804918	chr3:38557465	-1.906000
11	rs41310771	chr3:38577055	-0.609000
12	rs1805124	chr3:38603929	1.276000
13	rs7428779	chr3:38604932	-0.846000
14	rs41312433	chr3:38606151	3.844000
15	rs6770569	chr3:38606289	-2.353000
16	rs9873213	chr3:38608433	-0.438000
17	rs6797133	chr3:38614542	0.160000
18	rs6781731	chr3:38622216	-1.667000
19	rs7431305	chr3:38622673	-2.821000
20	rs11719543	chr3:38633499	-0.424000
21	rs41315485	chr3:38548784	1.757000

Next steps

- All downstream analyses will focus on protein-coding mutations, as these directly modify amino-acid sequence and can be evaluated with structural and biophysical metrics.
- **Noncoding variants require a different analytic strategy because their interpretation depends on *a priori* knowledge of regulatory architecture** (e.g., putative TF-binding motifs in promoters/CREs, miRNA-binding sites/MREs in 3′ UTRs, splice enhancers or silencers in intronic regions).
- These analyses can be performed bioinformatically, but they rely on an entirely different set of tools and databases, distinct from those used for protein-level variant assessment.

Pipeline for protein-coding mutants



Using reader_aminoConservation.py

- Description: Computes evolutionary conservation at each mutant amino-acid position by aligning the human protein sequence with its orthologs and scoring how conserved the reference residue is across species.
- Terminal:

```
python3 reader_aminoConservation.py -m
SCN5A_mutated_protein.fasta -g SCN5A -o
SCN5A_aminoConservation.tsv --run-mafft
```

- Where:
  - `-m` represents the FASTA file containing all mutant protein sequences plus the wild-type reference.
  - `-g` represents the human gene symbol used to retrieve orthologous protein sequences from Ensembl.
  - `-o` represents the output TSV file containing, for each variant, its amino-acid change, alignment column, conservation score, and cross-species residue counts.
  - `--run-mafft` performs the multiple sequence alignment automatically.

Using reader_aminoConservation.py

- Other outputs:
  - A fasta file containing all fetched orthologs from Ensembl
  - A fasta file containing the multiple sequence-aligned fasta files generated via MAFFT

SCN5A_orthologs.fasta

```
>ENSNLEG00000005433|nomascus_leucogenys
------------------------------------------------------------
>ENSPTRG00000014762|pan_troglodytes
MANFLLPRGTSSFRRFTRESLAAIEKRMAEKQARGSTTLQESREGLPEEEAPRPQLDLQASK
>ENSPPYG00000014004|pongo_abelii
MANFLLPRGTSSFRRFTRESLAAIEKRMVEKQARGSTTLQESREGLPEEEAPRPQLDLQASK
>ENSPPAG00000036665|pan_paniscus
MANFLLPRGTSSFRRFTRESLAAIEKRMAEKQARGSTTLQESREGLPEEEAPRPQLDLQASK
>ENSGGOG00000012938|gorilla_gorilla
MANFLLPRGTSSFRRFTRESLAAIEKRMAEKQARGSTTLQESREGLPKEEAPQPQLDLQASK
>ENSOANG00000051243|ornithorhynchus_anatinus
MAAAVLLPSGTHSFRRFTAESLAAIEKCAAAPRSRTGA-----HEPLPDPGEEPARPQVDLQA
>ENSSHAG00000022947|sarcophilus_harrisii
MLTVGLFVFSSCSYQGNWRPAPRKQDEKMANFLLPRGANSFRRFTRESLAAIEKQIAEKQAR-
>ENSMEUG00000006849|notamacropus_eugenii
MANLLLPRGTNSFRRFTRESLAAIEKQIAEKQAR--SITQESRDTPTEEEKPRPQLDLQASK
```

SCN5A_msa.fasta

```
>ENSNLEG00000005433|nomascus_leucogenys
------------------------------------------------------------
------------------------------------------------------------
----------------------------------TFIVLNKGKTIFRFSATNALYVLSP-
-------FHPIRRAAVKILV----HSL-FSMLIMCTILTNCVFMAQHDPPPWTKYVEYTF
TAIYTFESLVKILARGFCLHAFTFLRDPWNWLDFSVIIMAYTTEFVDLGNVSALRTFRVL
RALKTISVISGLKTIVGALIQSVKKLADVMVLTVFCLSVFALIGLQLFMGNLRHKCVRNF
TALNG---TN---GSVEA---DGLVWESLDLYLSDP-ENYLLKNGTSDVLLCGNSSDAGT
CPEGYRCLKAGENPDHGYTSFDSFAWAFLALFRLMTQDCWERLYQQTLRSAGKIYMIFFM
LVIFLGSFYLVNLILAVVAMAYEEQNQATIAETQEKEKRFQEAMEMLKKEHEALTIRGVD
TMSHSSLEMSPLASVNSHERRSKRRKRMSSGT-EECGEDRLPKSDSEDGPRAM-------
-----------------NHLSLTHSLSRTSMK-PRSSRGSIFTFRRRDLGSEADFADDEN
STAGESESHRTSLLVPWPLRRTSAQGQPSPGT--SAPGHALNGKKNSTVDCNGVVSLLGA
GN----PEATSPGSH-LLR--PVMLEH---PPDTTT-PSEEPG-GPQMVTSQAPCVDGFE
EPGARQRALSAVSVLTSALEELEESRRKCPPCWNRLAQRYLIWECCPLWTSIKQGVRLVV
MDPFADLTITMCIVLNTLFMALEHYNMTSEFEEMLQVGNLVFTGIFTAEMTFKIIALDPY
```

Using reader_aminoConservation.py

- Logic:
  - Reads the mutant + wild-type protein FASTA and automatically detects the gene symbol, reference transcript, and WT protein sequence based on previous outputs.
  - Fetches orthologous protein sequences for the same gene from Ensembl, since amino-acid conservation is not publicly catalogued and must be computed *de novo*.
  - Builds a combined FASTA containing all orthologs plus the human WT sequence, then generates a multiple sequence alignment (MSA) using MAFFT (if `--run-mafft` is specified).
  - Maps each mutant amino-acid position onto the aligned WT sequence to find its corresponding column in the MSA.
  - Computes per-site amino-acid conservation scores (entropy-based) across all orthologs, along with residue frequency distributions.
  - Outputs a TSV file listing, for every variant, its amino-acid change, alignment column, conservation score, and cross-species residue counts to support downstream variant prioritization.

Using reader_aminoConservation.py

- Scores range from 0 to 1, where higher values indicate stronger evolutionary constraint.
- A score near 1.0 reflects high conservation, suggesting the amino-acid position is structurally or functionally essential; substitutions here are more likely to affect channel gating, stability, or interactions.
- A score near 0.0 indicates evolutionary tolerance for variation, implying the site is less likely to drive pathogenic effects.
- Intermediate scores require contextual interpretation (domain location, physicochemical shifts, hotspot proximity).

SCN5A_aminoConservation.tsv

gene	rsid	aa_change	ref_aa	pos	alt_aa	ref_transcript	aln_col	cons_score	residue_counts
SCN5A	rs1805124	H558R	H	558	R	ENST00000423572	670	0.5139	H:16;Q:1;R:71

SCN10A_aminoConservation.tsv

gene	rsid	aa_change	ref_aa	pos	alt_aa	ref_transcript	aln_col	cons_score	residue_counts
SCN10A	rs6795970	V1073A	V	1073	A	ENST00000449082	2717	0.1982	A:29;D:3;G:22;K:1;L:2;P:7;Q:3;S:18;T:4;V:23;W:1
SCN10A	rs7374804	K950N	K	950	N	ENST00000449082	2394	0.6560	A:2;C:1;D:2;E:9;G:12;H:1;K:194;M:1;N:4;P:1;Q:5;R:5;
SCN10A	rs7630989	S509A	S	509	A	ENST00000449082	1371	0.2457	A:74;E:1;F:4;G:2;I:16;L:44;M:3;P:38;Q:1;R:7;S:19;T:
SCN10A	rs74717885	I206M	I	206	M	ENST00000449082	471	0.3967	E:1;H:1;I:74;K:3;M:3;R:2;S:15;T:36;V:109;Y:1

Using reader_aminoConservation.py

- Download MAFFT: https://mafft.cbrc.jp/alignment/software/macosx.html
- Note: Because no public database provides per-residue amino-acid conservation scores for every human gene, these values must be computed locally (*de novo*) by fetching orthologs manually and constructing a multiple sequence alignment.

Using reader_uniprot.py

- Description: Maps each mutant amino-acid position onto UniProtKB-annotated protein features to provide structural and functional context for every variant (e.g., transmembrane helices, cytoplasmic domains, disordered regions, PTM sites, “natural variant” annotations). Uses the UniProt REST API and the mutant FASTA headers to generate a TSV table linking rsIDs to UniProt features.
- Terminal:

```
python3 reader_uniprot.py -m SCN10A_mutated_protein.fasta -g SCN10A
-u Q9Y5Y9 -o SCN10A_uniprot_mapping_Q9Y5Y9.tsv
```

- Where:
  - `-m` represents the input FASTA filename containing all mutant protein sequences plus the wild-type entry.
  - `-g` represents the approved gene symbol of interest (e.g., SCN10A).
  - `-u` represents the UniProt accession corresponding to the canonical protein isoform (e.g., Q9Y5Y9 for Nav1.8).
  - `-o` represents the output TSV filename listing rsID, amino-acid change, protein position, UniProt feature type/description, and feature range that overlap the mutated residue.

Using reader_uniprot.py

- Logic:
  - Reads a mutant + wild-type protein FASTA, using standardized headers to extract the rsID and amino-acid change (e.g., V1073A) for each variant.
  - Contacts the UniProt REST API once to download all feature annotations for the specified accession (domains, repeats, topological regions, disordered segments, PTM sites, natural variants, etc.).
  - For each mutant entry (WT is skipped), parses the amino-acid position and identifies all UniProt features whose coordinate range includes that residue.
  - Writes a TSV file in which each row links an rsID to its HGVS protein notation (p.XnY), protein position, UniProt accession, and any overlapping UniProt features, thereby **indicating whether the mutation falls in a transmembrane helix, cytoplasmic loop, disordered region, regulatory motif, or previously known variant site.**

Using reader_uniprot.py

SCN5A_uniprot_mapping_Q14524.tsv

rsID	gene	hgvsp	aa_change	protein_pos	uniprot_acc	uniprot_feature_type	uniprot_feature_desc	uniprot_feature_
rs1805124	SCN5A	p.H558R	H558R	558	Q14524	Chain	Sodium channel protein type 5 subunit alpha 1-2016	
rs1805124	SCN5A	p.H558R	H558R	558	Q14524	Topological domain	Cytoplasmic 414-719	
rs1805124	SCN5A	p.H558R	H558R	558	Q14524	Region	Disordered 461-591	
rs1805124	SCN5A	p.H558R	H558R	558	Q14524	Natural variant	channels properties are similar to wild-type; the double mu	

Using reader_pathogenicity.py

- Description: Annotates missense mutants with clinical and in silico pathogenicity predictions by combining information from NCBI ClinVar/dbSNP and Ensembl VEP into a single TSV file. The script reads a FASTA file of wild-type and mutant protein sequences, extracts rsIDs and amino-acid changes from standardized headers, skips the wild-type entry, and queries online resources for each mutant. It summarizes ClinVar clinical significance (with a simple numeric score) and VEP-derived predictions (impact, consequence terms, SIFT, PolyPhen) so that every variant has a harmonized pathogenicity profile.
- Terminal:

```
python3 reader_pathogenicity.py -i SCN5A_mutated_protein.fasta -o
SCN5A_predictions.tsv
```

- Where:
  - `-i` represents the input FASTA file containing wild-type and mutant protein sequences with headers of the form `GENE|rsID|AA_CHANGE|mut_protein|TRANSCRIPT` and `GENE|WT|TRANSCRIPT|` wildtype_protein.
  - `-o` represents the output TSV file that compiles, for each mutant, its gene, amino-acid change, rsID, summarized ClinVar class/score, and Ensembl VEP impact, consequence terms, SIFT prediction, and PolyPhen prediction (plus raw ClinVar counts in JSON format).
  - `--delay` (optional, default 1.0) sets the pause in seconds between successive API calls to avoid overwhelming NCBI/VEP servers.

Using reader_pathogenicity.py

- Logic:
  - Parses the mutant + wild-type FASTA file, using the header structure to extract the gene symbol, rsID, and amino-acid change for each entry.
  - Automatically skips the wild-type record, ensuring that only mutant entries are annotated.
  - For each rsID-bearing mutant, queries NCBI dbSNP/ClinVar via the Variation API, collects all reported clinical significance labels, normalizes them, and converts them into a simple quantitative score with a dominant class (e.g., pathogenic, likely benign, uncertain).
  - Optionally queries Ensembl VEP for the same rsID to retrieve impact assignments, consequence terms, and algorithm-specific predictions (SIFT, PolyPhen).
  - Handles variants without rsIDs or without ClinVar/VEP data by labeling them appropriately (`no_rsid`, `no_data`, or `NA`) while still writing a row.
  - Outputs a single TSV in which each mutant variant is represented by one line **summarizing all available pathogenicity evidence** (at least the ones with APIs)**, ready for downstream integration into the variant-ranking pipeline.**

Using reader_pathogenicity.py

- Pathogenicity integrates two independent evidence streams:
  (1) ClinVar clinical significance, reflecting human expert or submitter consensus;
  (2) VEP computational predictions (impact, consequence terms, SIFT, PolyPhen), reflecting in silico estimates of functional disruption.
- ClinVar score (0–1.0) represents a weighted average of all significance labels found for the variant:
  - 1.0 = consistently reported as pathogenic
  - 0.8 = predominantly likely pathogenic
  - 0.5 = uncertain significance
  - 0.2 = predominantly likely benign
  - 0.0 = consistently benign

SCN10A_pathogenicityScore.tsv

gene	aa_change	rsid	clinvar_class	clinvar_score	vep_impact	vep_consequences	vep_sift	vep_polyphen	cli
SCN10A	V1073A	rs6795970	benign	0.000	MODERATE	missense_variant	tolerated_low_confidence	benign	{"benign":
SCN10A	K950N	rs7374804	benign	0.200	MODERATE	missense_variant	deleterious_low_confidence	probably_damaging	
SCN10A	S509A	rs7630989	benign	0.067	MODERATE	missense_variant	tolerated_low_confidence	benign	{"likely_ben
SCN10A	I206M	rs74717885	benign	0.067	MODERATE	missense_variant	deleterious_low_confidence	benign	{"likely_ben

Using reader_pathogenicity.py

- VEP impact communicates predicted severity:
  - HIGH (e.g., stop-gained)
  - MODERATE (missense)
  - LOW (synonymous)
  - MODIFIER (noncoding)
- SIFT and PolyPhen provide residue-specific estimates of functional disruption:
- SIFT: deleterious vs tolerated
- PolyPhen: probably damaging, possibly damaging, benign

SCN10A_pathogenicityScore.tsv

gene	aa_change	rsid	clinvar_class	clinvar_score	vep_impact	vep_consequences	vep_sift	vep_polyphen	cli
SCN10A	V1073A	rs6795970	benign	0.000	MODERATE	missense_variant	tolerated_low_confidence	benign	{"benign": 4
SCN10A	K950N	rs7374804	benign	0.200	MODERATE	missense_variant	deleterious_low_confidence	probably_damaging	
SCN10A	S509A	rs7630989	benign	0.067	MODERATE	missense_variant	tolerated_low_confidence	benign	{"likely_ben
SCN10A	I206M	rs74717885	benign	0.067	MODERATE	missense_variant	deleterious_low_confidence	benign	{"likely_ben

Using reader_aaProperties.py

- Description: Computes, for each mutant, how drastic the amino-acid substitution is relative to the wild-type residue using standard physicochemical scales. The script reads a FASTA file containing mutant + WT protein sequences, parses the mutation code from the header (e.g., H558R), and calculates changes in charge, polarity class, side-chain volume, Kyte–Doolittle hydropathy, helix propensity (Chou–Fasman), and Grantham distance. Results are written to a tab-separated file (e.g., SCN5A_aaProperties.tsv) for integration into the variant-ranking pipeline.
- Terminal:

```
python3 reader_aaProperties.py -i
SCN5A_mutated_protein.fasta -o SCN5A_aaProperties.tsv
```

- Where:
  - `-i` is the input FASTA containing mutant + wild-type protein sequences.
  - `-o` is the TSV output file summarizing physicochemical properties and their WT→mutant deltas for each variant.

Using reader_aaProperties.py

- Logic:
  - Reads the mutant + WT FASTA, identifies the wild-type entry (wildtype_protein), and stores its reference sequence.
  - For each mutant header, extracts the mutation code (e.g., H558R) and position, and sanity-checks that the WT sequence truly has the expected reference residue at that position.
  - Looks up per–amino-acid properties from predefined dictionaries: net charge, polarity class, side-chain volume, hydropathy (Kyte–Doolittle), and helix propensity (Chou–Fasman).
  - Computes WT and mutant values plus their differences (Δcharge, Δvolume, Δhydropathy, Δhelix) and the Grantham distance between the two residues.
  - Writes one TSV row per variant containing the gene, rsID, mutation, position, WT/mutant properties, and all computed deltas to quantify how “biophysically disruptive” each substitution is.

Using reader_aaProperties.py

- Charge change (Δcharge)
  - Possible range: –1 to +1 (occasionally ±2 in rare cases).
  - Large shifts (e.g., +1 → –1) are often disruptive, especially in transmembrane helices, voltage-sensor segments, pore loops, or regions mediating electrostatic interactions.
  - Promising variants show a gain or loss of charge in domains known to be functionally sensitive (e.g., S4 gating charges).
- Polarity class change
  - Categories: nonpolar → polar → acidic/basic.
  - A shift across classes (e.g., nonpolar → charged) suggests altered side-chain chemistry affecting packing, solvent exposure, or membrane embedding.
  - Class flips are often high-priority candidates.

Using reader_aaProperties.py

- Volume change (Δvolume)
  - Typical range: –100 to +100 Å³.
  - Δvolume > 30–40 Å³ may indicate steric clashes or cavity formation.
  - Particularly informative in tight structural elements (helices, pore loops).
- Hydropathy change (ΔKD)
  - Kyte–Doolittle scale: –4.5 to +4.5.
  - ΔKD > 1.0 suggests substantial alteration in local hydrophobicity, which is especially relevant for transmembrane segments.
  - Negative shifts (hydrophobic → hydrophilic) in membrane regions often destabilize folding or insertion.

Using reader_aaProperties.py

- Helix propensity change (Δhelix)
  - Chou–Fasman values: ~0.5–1.5.
  - Δhelix > 0.3 may weaken or strengthen α-helix formation and affect gating helices (S1–S6).
  - Loss of helix propensity in predicted TM helices is commonly pathogenic.
- Grantham distance
  - Range: 0–215.
  - <50: conservative substitution (usually tolerated).
  - 50–100: moderately radical.
  - >100: highly radical; substantial chemical dissimilarity.
  - >150: strongly disruptive; high priority for follow-up.

Using reader_aaProperties.py

SCN10A_aaProperties.tsv

gene	rsid	mutation	position	wt_aa	mut_aa	charge_wt	charge_mut	delta_charge	polarity_wt	polarity_mut	
SCN10A	rs6795970	V1073A	1073	V	A	0	0	0	nonpolar	nonpolar	140.0	88.6	-51.400000000000006	4.2	1.8
SCN10A	rs7374804	K950N	950	K	N	1	0	-1	basic	polar	168.6	114.1	-54.5	-3.9	-3.5	0.39999999999999
SCN10A	rs7630989	S509A	509	S	A	0	0	0	polar	nonpolar	89.0	88.6	-0.4000000000000057	-0.8	1.8	2.6
SCN10A	rs74717885	I206M	206	I	M	0	0	0	nonpolar	nonpolar	166.7	162.9	-3.799999999999983	4.5	1.9	-2.6

Conclusion

- We established a modular Python pipeline that integrates coding consequences, conservation metrics, UniProt features, physicochemical changes, and pathogenicity predictions to systematically prioritize variants.
- Python provides reliable access to public APIs and scalable sequence-processing tools, making the workflow reproducible and eliminating manual annotation.
- The design is fully extensible to other sodium-channel genes and relevant transcription factors, ensuring backward compatibility as additional Brugada-associated loci are identified in the project.
- This framework delivers a transparent, data-driven foundation that guides experimental validation and complements clinical interpretation.