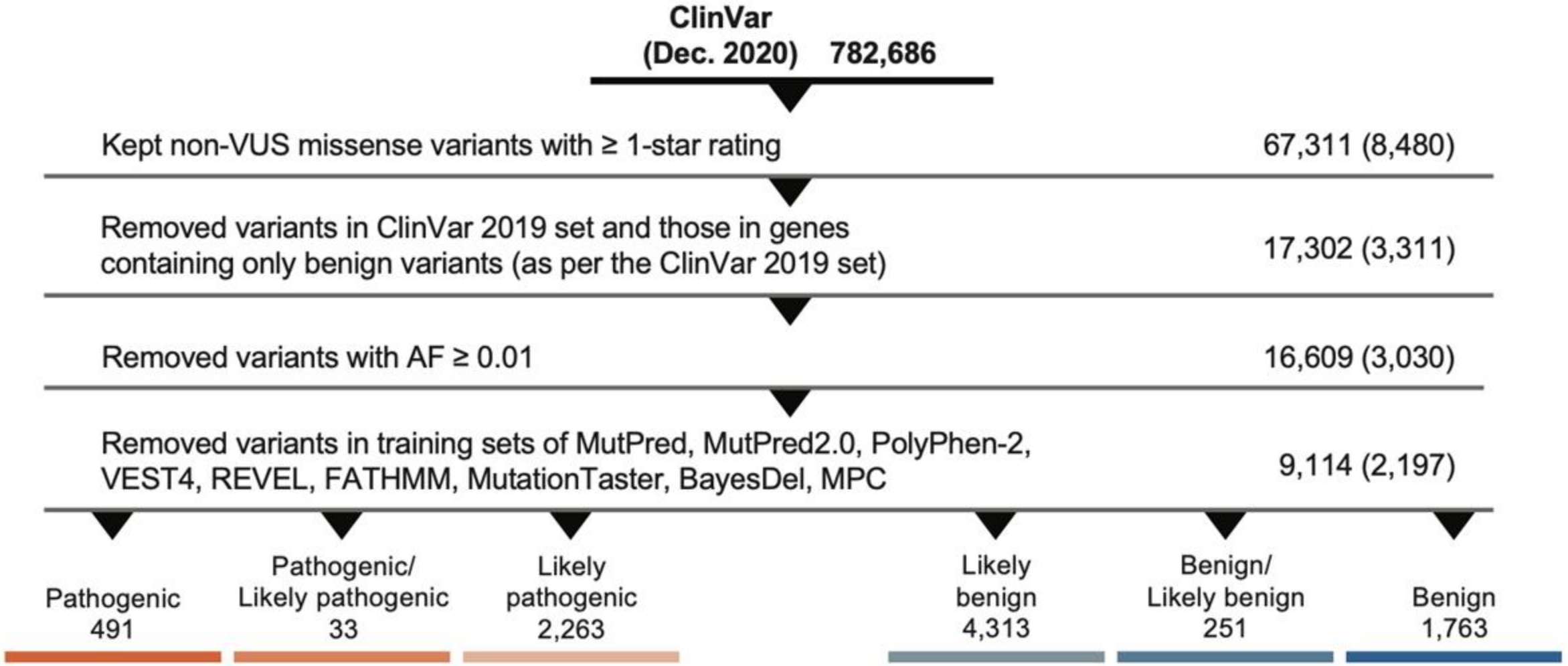
1. ClinVar

Variants from Clinvar were downloaded from its FTP site in 02.09.2024

Index of /pub/clinvar/tab_delimited

Name	Size	Released	Last Modified
Parent Directory	-		
supplements/	-		2021-12-03 09:41:31
special_requests/	-		2021-12-03 09:41:31
archive/	-		2024-09-05 00:05:29
allele_gene.txt.gz	74,483,633	2024-03-31 05:31:50	2024-09-02 11:35:58
allele_gene.txt.gz.md5	87	2024-03-31 19:46:12	2024-09-03 09:45:32
cross_references.txt	57,750,772	2024-03-31 04:43:28	2024-09-02 11:28:58
cross_references.txt.md5	89	2024-03-31 19:46:01	2024-09-03 09:45:27
gene_specific_summary.txt	3,434,835	2024-03-31 04:45:50	2024-09-02 11:29:34
gene_specific_summary.txt.md5	94	2024-03-31 19:46:01	2024-09-03 09:45:23
hgvs4variation.txt.gz	363,504,806	2024-03-31 05:04:34	2024-09-02 11:53:19
hgvs4variation.txt.gz.md5	90	2024-03-31 19:46:09	2024-09-03 09:45:27
organization_summary.txt	739,032	2024-03-31 04:52:33	2024-09-02 11:39:42
organization_summary.txt.md5	93	2024-03-31 19:46:06	2024-09-03 09:45:28
README	47,034	2024-03-07 04:06:55	2024-06-03 08:57:04
submission_summary.txt.gz	269,429,898	2021-11-30 02:41:14	2024-09-02 12:05:59
submission_summary.txt.gz.md5	94	2024-03-31 19:46:08	2024-09-03 09:45:34
summary_of_conflicting_interpretations.txt	1,196,606,376	2021-11-30 02:39:52	2024-09-02 11:47:39
summary_of_conflicting_interpretations.txt.md5	111	2024-03-31 19:46:21	2024-09-03 09:45:37
var_citations.txt	175,268,185	2024-03-31 04:44:43	2024-09-02 11:50:12
var_citations.txt.md5	86	2024-03-31 19:46:05	2024-09-03 09:45:37
variant_summary.txt.gz	**282,753,188**	**2024-03-31 19:30:16**	**2024-09-03 09:10:47**
variant_summary.txt.gz.md5	91	2024-03-31 19:46:01	2024-09-03 09:45:33
variation_allele.txt.gz	16,347,814	2024-03-31 05:03:31	2024-09-02 11:36:37
variation_allele.txt.gz.md5	92	2024-03-31 19:46:11	2024-09-03 09:45:31

Figure 1, Pejaver et al. 2022



Pejaver et al. 2022 study filters:

- Only missense variants with an allele frequency (AF) below 0.01 in the Genome Aggregation Database (gnomAD v.2.1) were first retained
- Genes with at least one pathogenic variant of any type in ClinVar were first retained
- For each variant, the gnomAD exomes global AF was used. When this was unavailable, the gnomAD genomes global AF was used.
- all VUSs, variants with a zero-star review status, i.e., without any detailed review information, and those with conflicting classifications were excluded.

Additional filter for our study:

- Keep submissions from 2021 to 2024

1) Filter for single-nucleotide variants

```
clinvar_data['Type'].unique()
```

```
array(['Indel', 'Deletion', 'single nucleotide variant', 'Duplication',
       'Microsatellite', 'Insertion', 'Variation', 'Complex',
       'Translocation', 'Inversion', 'copy number gain', 'fusion',
       'copy number loss', 'protein only', 'Tandem duplication'],
      dtype=object)
```

```
clinvar_data_filter=clinvar_data[clinvar_data.Type =='single nucleotide variant'].reset_index(drop=True)
len(clinvar_data_filter)
```

```
5498097
```

Filter: Remove if there is no “missense variant” information (NaNs, Ter, etc.)

```
# Function to check if both parts of the variant are valid amino acids
def is_valid_variant(variant_3letter):
    valid_amino_acids = set(three_to_one.keys())  # Get the set of valid three-letter amino acids

    if pd.isna(variant_3letter):  # Handle NaN values
        return False
    # Extract the three-letter amino acid codes (first 3 and last 3 characters)
    three_letter_from = variant_3letter[:3]
    three_letter_to = variant_3letter[-3:]

    # Check if both amino acids are in the valid set and no "Ter" (stop codon)
    if (three_letter_from in valid_amino_acids and three_letter_to in valid_amino_acids
        and 'Ter' not in variant_3letter):
        return True
    else:
        return False
```

```
# Apply the filter function to the 'variant_3letter' column
clinvar_data_filter_clean = clinvar_data_filter[clinvar_data_filter['variant_3letter'].apply(is_valid_variant)]
```

2) Keep submissions from 2021 and later

```
# Filter rows where 'LastEvaluated_year' is 2021 or later
filtered_from_2021 = clinvar_data_filter_clean[clinvar_data_filter_clean['LastEvaluated_year'] >= 2021]
```

3) Exclude Variants with Zero-Star Review Status, VUS, and Conflicting Classifications

- remove vus, conflicting

```
filtered_from_2021.ClinicalSignificance.unique()
```

```
array(['Conflicting classifications of pathogenicity',
       'Pathogenic/Pathogenic, low penetrance; other; risk factor',
       'Pathogenic/Likely pathogenic/Pathogenic, low penetrance; other',
       'Uncertain significance', 'Pathogenic/Likely pathogenic',
       'Pathogenic', 'Likely benign', 'Likely pathogenic', 'Benign',
       'Benign/Likely benign',
       'Conflicting classifications of pathogenicity; risk factor',
       'drug response',
       'Conflicting classifications of pathogenicity; association; risk factor',
       'Benign/Likely benign; other',
       'Conflicting classifications of pathogenicity; other',
       'Pathogenic/Likely pathogenic; risk factor', 'not provided',
       'Likely benign; other', 'Pathogenic; risk factor',
       'Conflicting classifications of pathogenicity; association',
       'Benign/Likely benign; association',
       'Benign/Likely benign; other; risk factor',
       'Conflicting classifications of pathogenicity; protective',
       'Pathogenic; drug response',
       'Pathogenic/Likely pathogenic/Pathogenic, low penetrance',
       'Uncertain significance/Uncertain risk allele',
       'Pathogenic/Likely pathogenic; other',
       'Likely pathogenic; association',
       'Uncertain significance; drug response',
       'Likely pathogenic; drug response',
       'Pathogenic/Likely risk allele',
       'Pathogenic/Likely pathogenic/Likely risk allele',
       'Conflicting classifications of pathogenicity; other; risk factor',
       'other', 'Pathogenic; other',
       'Pathogenic/Pathogenic, low penetrance; other', 'Benign; other',
       'Likely risk allele', 'Uncertain risk allele; risk factor',
       'Benign; drug response', 'Likely pathogenic/Likely risk allele',
       'Uncertain significance; other',
       'no classifications from unflagged records', 'risk factor',
       'Conflicting classifications of pathogenicity; Affects',
       'Pathogenic; association', 'Affects',
       'Likely pathogenic; risk factor', 'Likely benign; association',
       'Benign; association', 'Uncertain significance; risk factor',
       'Uncertain significance; association', 'association', 'protective',
       'Likely pathogenic, low penetrance'], dtype=object)
```

```
filtered_from_2021_clean = filtered_from_2021[filtered_from_2021['ClinicalSignificance'].isin(
                ['Pathogenic','Likely pathogenic','Pathogenic/Likely pathogenic',
                 'Benign', 'Likely benign', 'Benign/Likely benign'])]
```

```
[31]: filtered_from_2021_clean.ReviewStatus.value_counts()
```

```
t[31]: criteria provided, single submitter                    168526
       criteria provided, multiple submitters, no conflicts    83378
       no assertion criteria provided                           4540
       reviewed by expert panel                                 3706
       Name: ReviewStatus, dtype: int64
```

```
[32]: # Keep only rows where the review status is not zero-star
      filtered_from_2021_clean_reviewed = filtered_from_2021_clean[filtered_from_2021_clean['ReviewStatus'] != 'no asserti
```

Filter: Remove if there is no Chromosome info

```
filtered_from_2021_clean_reviewed.Chromosome.unique()
```

```
array(['6', '10', '16', '22', '15', '7', '1', '8', '21', '5', '11', '19',
       '4', '3', '17', '12', '20', '9', '18', '2', '14', '13', 'Y', 'X',
       'na', 2, 1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 'Un'], dtype=object)
```

Keep missense vartiants with an allele frequency (AF) below 0.01 in the Genome Aggregation Database (gnomAD v.2.1)

at this point. we need to pass the data through VEP in order to get gnomAD information

first: handle the ASSEMBLY ISSUE

```
filtered_from_2021_clean_reviewed2=pd.read_csv('../data/clinvar/filtered_from_2021_clean_reviewed2.csv')
```

```
filtered_from_2021_clean_reviewed2.Assembly.value_counts()
```

```
GRCh37    127763
GRCh38    127761
Name: Assembly, dtype: int64
```

Filter: Several variants have the same value in “AlleleID” column. The only difference is the Assembly. Drop duplicates

```
filtered_from_2021_clean_reviewed3=filtered_from_2021_clean_reviewed2.drop_duplicates(subset=['#AlleleID'],keep='fir
filtered_from_2021_clean_reviewed3.Assembly.value_counts()
```

```
GRCh38    127741
GRCh37         7
Name: Assembly, dtype: int64
```

Separate them before using with VEP (for retrieving gnomAD info). Coordinates info should not be mixed

1. Prepare input file
   file_grch38.vcf, file_grch37.vcf
2. run VEP to retrieve gnomAD info
3. merge the output with the original file

Filter: gnomAD frequency

```
# Convert gnomADe_AF and gnomADg_AF to numeric, coercing errors
cleaned_missense_df['gnomADe_AF'] = pd.to_numeric(cleaned_missense_df['gnomADe_AF'], errors='coerce')
cleaned_missense_df['gnomADg_AF'] = pd.to_numeric(cleaned_missense_df['gnomADg_AF'], errors='coerce')

# Create a new column 'gnomAD_AF' that fills missing exomes AF with genomes AF
cleaned_missense_df['gnomAD_AF'] = cleaned_missense_df['gnomADe_AF'].fillna(cleaned_missense_df['gnomADg_AF'])

# Filter for variants where gnomAD_AF < 0.01 and drop rows where gnomAD_AF is NaN
filtered_df = cleaned_missense_df[cleaned_missense_df['gnomAD_AF'] < 0.01].dropna(subset=['gnomAD_AF'])
```

Keep Genes with at least one pathogenic variant of any type in ClinVar

```
filtered_df.ClinicalSignificance.unique()
```

```
array(['Pathogenic', 'Benign/Likely benign',
       'Pathogenic/Likely pathogenic', 'Likely benign', 'Benign',
       'Likely pathogenic'], dtype=object)
```

```
def filter_pathogenic_genes(df):
    # Define pathogenic-related terms
    pathogenic_terms = ['Pathogenic', 'Likely pathogenic', 'Pathogenic/Likely pathogenic']

    # Filter for genes that have at least one pathogenic variant
    pathogenic_genes = df[df['ClinicalSignificance'].isin(pathogenic_terms)]['GeneSymbol'].unique()

    # Filter the DataFrame to retain only rows for those genes
    df_filtered = df[df['GeneSymbol'].isin(pathogenic_genes)]

    return df_filtered
```

```
filtered_df2 = filter_pathogenic_genes(filtered_df1)
```

```
len(filtered_df2)
```

```
48043
```

 # To do: we need to filter for canonicals only. But, we will do it after getting the in silico tool predictions

Predictor: VEST4 (we run it using the tool's page because it is not inside the VEP's tools list)

http://cravat.us/CRAVAT/

Note: the CRAVAT/VEST4 tool works well if you create an account

1. VCF file prepare 2. Run VEST4 3. Parse the output to merge with the original file

Potential problems that need to be filtered: The input file might result in different variants, in this case we eliminate them, bcs we prioritize whatever coordinates & variants we have in the clinvar

---

Predictors: VEP output

```
vep_out_all = pd.read_csv('../data/clinvar/vep_out_clinvar.txt', sep='\t')
vep_out_all2 = pd.read_csv('../data/clinvar/vep_out_missing.txt', sep='\t')
```

Potential problems: All input might not be parsed. In that case we either eliminate them or we run the tool again? Perhaps the web-based version is more problematic than command-line version?

Parse the predictors & obtain binary predictions

Potential problems: predictors have different formats, each case handled separately or by grouping them in similar functions
Check ***VEP.py*** to see the functions if you want to explore

Thresholds source: check the ***Thresholds_log.xlxs*** file and the related links like https://www.varianteffect.org/veps

am_pathogenicity	am_class
0.3314	likely_benign
0.2498	likely_benign
0.37	ambiguous
0.37	ambiguous
0.9956	likely_pathogenic
...	...
0.0957	likely_benign
0.0964	likely_benign
0.0667	likely_benign
0.2436	likely_benign

REVEL
0.902
0.869
0.700
0.700
0.930
...
0.200
0.319
0.378

MutationTaster_score	MutationTaster_pred
1,1,1,1	D,D,D,D
1,1,0.999998,1	D,D,D,D
1,1,1,1	D,D,D,D
1,1,1,1	D,D,D,D
0.99999,0.99999,0.999999,0.99999	D,D,D,D
...	...
1	N
1	N
0.999971	N
0.913325	N
0.907747	D

Parse the predictors & obtain binary predictions

With prints of column names used & created

```
df = VEP.clean_vep_data(vep_out_all_merged)
```

```
SIFT // SIFT_cat // SIFT_quant // SIFT_binary

PolyPhen // PolyPhen_cat // PolyPhen_quant // PolyPhen_binary

CADD_PHRED // CADD_PHRED_parsed // CADD_PHRED_binary_(thr_P>=19)

REVEL // REVEL_parsed // REVEL_binary_(thr_P>0.5)

ClinPred // ClinPred_parsed // ClinPred_binary_(thr_P>=0.5)

Eigen-PC-raw_coding // Eigen-PC-raw_coding_parsed // Eigen-PC-raw_coding_binary_(thr_P>0)

Eigen-raw_coding // Eigen-raw_coding_parsed // Eigen-raw_coding_binary_(thr_P>=0)

GERP++_RS // GERP++_RS_parsed // GERP++_RS_binary_(thr_P>=2)

DANN_score // DANN_score_parsed // DANN_score_binary_(thr_P>=0.99)

MVP_score // MVP_score_parsed // MVP_score_binary_(thr_P>0.7)

BayesDel_noAF_score // BayesDel_noAF_pred // BayesDel_noAF_score_parsed // BayesDel_noAF_pred_binary

am_pathogenicity // am_class // am_pathogenicity_parsed // am_class_binary

EVE_SCORE // EVE_CLASS // EVE_SCORE_parsed // EVE_CLASS_binary

MetaLR_score // MetaLR_pred // MetaLR_score_parsed // MetaLR_pred_binary

MetaSVM_score // MetaSVM_pred // MetaSVM_score_parsed // MetaSVM_pred_binary

LRT_score // LRT_pred // LRT_score_parsed // LRT_pred_binary

M-CAP_score // M-CAP_pred // M-CAP_score_parsed // M-CAP_pred_binary

PrimateAI_score // PrimateAI_pred // PrimateAI_score_parsed // PrimateAI_pred_binary

MutPred_score // MutPred_score_parsed // MutPred_score_binary_(thr_P>=0.5)

VARITY_R_score // VARITY_R_score_parsed // VARITY_R_score_binary_(thr_P>=0.5)

MPC_score // MPC_score_parsed // MPC_score_binary_(thr_P>=2)

gMVP_score // gMVP_score_parsed // gMVP_score_binary_(thr_P>=0.75)

FATHMM_pred // FATHMM_score // FATHMM_pred_parsed  // FATHMM_score_parsed  // FATHMM_pred_binary

MutationTaster_pred // MutationTaster_score // MutationTaster_pred_parsed  // MutationTaster_score_parsed  // Mutat
ionTaster_pred_binary

MutationAssessor_pred // MutationAssessor_score // MutationAssessor_pred_parsed  // MutationAssessor_score_parsed
// MutationAssessor_pred_binary

ESM1b_pred // ESM1b_score // ESM1b_pred_parsed  // ESM1b_score_parsed  // ESM1b_pred_binary

MetaRNN_pred // MetaRNN_score // MetaRNN_pred_parsed  // MetaRNN_score_parsed  // MetaRNN_pred_binary

PROVEAN_pred // PROVEAN_score // PROVEAN_pred_parsed  // PROVEAN_score_parsed  // PROVEAN_pred_binary

DEOGEN2_pred // DEOGEN2_score // DEOGEN2_pred_parsed  // DEOGEN2_score_parsed  // DEOGEN2_pred_binary

LIST-S2_pred // LIST-S2_score // LIST-S2_pred_parsed  // LIST-S2_score_parsed  // LIST-S2_pred_binary
```

Merge with the original clinvar file based on some common columns

Again, some variants will be lost bcs the same chr, ref allele, alt allele do not correspond to same amino acid variant in the output file of VEP, so I did an inner merge this time

```
merged_inner = clinvar_to_merge.merge(df_to_merge3, on=merge_cols, how='inner')
print(len(merged_inner))

check_coverage(merged_inner)
```

```
47606
```

Filter for taking canonical only (bcs all our features are calculated for the canonical sequence & structure)

Originally, Clinvar does not include a straightforward info related to uniprot / canonical information

```
1 #AlleleID
2 Type
3 Name
4 GeneID
5 GeneSymbol
6 HGNC_ID
7 ClinicalSignificance
8 ClinSigSimple
9 LastEvaluated
10 RS# (dbSNP)
11 nsv/esv (dbVar)
12 RCVaccession
13 PhenotypeIDS
14 PhenotypeList
15 Origin
16 OriginSimple
17 Assembly
18 ChromosomeAccession
19 Chromosome
20 Start
21 Stop
22 ReferenceAllele
23 AlternateAllele
24 Cytogenetic
25 ReviewStatus
26 NumberSubmitters
27 Guidelines
28 TestedInGTR
29 OtherIDs
30 SubmitterCategories
31 VariationID
32 PositionVCF
33 ReferenceAlleleVCF
34 AlternateAlleleVCF
35 SomaticClinicalImpact
36 SomaticClinicalImpactLastEvaluated
37 ReviewStatusClinicalImpact
38 Oncogenicity
39 OncogenicityLastEvaluated
40 ReviewStatusOncogenicity
```

But we have these three columns coming from VEP tool after merging

```
#Uploaded_variation
Location
Allele
Consequence
IMPACT
SYMBOL
Gene
Feature_type
Feature
BIOTYPE
EXON
INTRON
HGVSc
HGVSp
cDNA_position
CDS_position
Protein_position
Amino_acids
Codons
Existing_variation
REF_ALLELE
UPLOADED_ALLELE
DISTANCE
STRAND
FLAGS
SYMBOL_SOURCE
HGNC_ID
MANE
MANE_SELECT
MANE_PLUS_CLINICAL
TSL
APPRIS
ENSP
SWISSPROT
TREMBL
UNIPARC
UNIPROT_ISOFORM
SIFT
PolyPhen
AF
Uniprot_acc
Uniprot_entry
VARITY_ER_LOO_rankscore
VARITY_ER_LOO_score
VARITY_ER_rankscore
VARITY_ER_score
VARITY_R_LOO_rankscore
VARITY_R_LOO_score
VARITY_R_rankscore
VARITY_R_score
VindijiaNeandertal
aapos
bStatistic
bStatistic_converted_rankscore
clinvar_MedGen_id
clinvar_OMIM_id
clinvar_Orphanet_id
```

```
merged_inner[['GeneSymbol','Feature','Uniprot_acc', 'Uniprot_entry','UNIPROT_ISOFORM']]
```

	GeneSymbol	Feature	Uniprot_acc	Uniprot_entry	UNIPROT_ISOFORM
0	MYO15A	ENST00000647165.2	A0A087WYA1,Q9UKN7,Q9UKN7	A0A087WYA1_HUMAN,MYO15_HUMAN,MYO15_HUMAN	Q9UKN7-1
1	NOTCH1	ENST00000651671.1	P46531	NOTC1_HUMAN	-
2	GALT	ENST00000378842.8	P07902-2,P07902	GALT_HUMAN,GALT_HUMAN	P07902-1
3	KCNV2	ENST00000382082.4	Q8TDN2	KCNV2_HUMAN	-
4	CSPP1	ENST00000678616.1	Q1MSJ5-1,Q1MSJ5-2	CSPP1_HUMAN,CSPP1_HUMAN	-
...	...	...	...	...	...
47601	CENPJ	ENST00000381884.9	Q9HC77-2,Q9HC77,F6VUX8	CENPJ_HUMAN,CENPJ_HUMAN,F6VUX8_HUMAN	Q9HC77-1
47602	SEMA6B	ENST00000586582.6	Q9H3T3,Q9H3T3-3	SEM6B_HUMAN,SEM6B_HUMAN	Q9H3T3-1
47603	KCNT1	ENST00000371757.7	Q5JUK3-2,Q5JUK3-4,Q5JUK3-3,C9J9Y7,A0A0D9SFC8,A...	KCNT1_HUMAN,KCNT1_HUMAN,KCNT1_HUMAN,C9J9Y7_HUM...	Q5JUK3-3
47604	EGLN1	ENST00000366641.4	Q9GZT9	EGLN1_HUMAN	Q9GZT9-1
47605	ATP1A2	ENST00000361216.8	P50993,B1AKY9	AT1A2_HUMAN,B1AKY9_HUMAN	-

```python
def process_and_harmonize_data(df):
    """
    Complete pipeline to process protein data:
    1. Identify canonical entries
    2. Filter for canonical proteins
    3. Filter for majority features
    4. Harmonize remaining UniProt discrepancies

    Parameters:
    df: DataFrame with required columns:
        UNIPROT_ISOFORM, Uniprot_acc, GeneSymbol, Features

    Returns:
    DataFrame: Processed and harmonized DataFrame
    """
```

This pipeline:

1. Identifies and filters for canonical proteins
2. Standardizes UniProt IDs
3. Filters for majority of “Feature” column
4. Harmonizes any remaining UniProt discrepancies
5. Provides detailed statistics at each step
6. Performs final validation to ensure unique genes match unique UniProt IDs

The final result should give you a clean dataset where:

- Only canonical proteins are included
- Each gene has consistent features
- Each gene maps to exactly one UniProt ID

```python
# Process the data through all filters
clin_final = process_and_harmonize_data(merged_inner)
```

```
Initial statistics:
Total rows: 47606
Unique genes: 2027
Unique UniProt IDs: 5225

After canonical filtering:
Remaining rows: 38764

After feature filtering:
Remaining rows: 38264

Final statistics:
Final rows: 38264
Unique genes: 1860
Unique UniProt IDs: 1860
```

```python
print(clin_final.BinaryClinicalSignificance.value_counts())
print(len(clin_final))
print(clin_final.uniprot.nunique())
print(clin_final['GeneSymbol'].nunique())
```

```
B    27109
P    11155
Name: BinaryClinicalSignificance, dtype: int64
38264
1860
1860
```

ClinVar

FINAL DATASET

- 2105 unique genes
- 46,050 variants
    - 32,051 Benign
    - 13,999 Pathogenic

- Code cleaning & documenting DONE
- Obtaining all predictions & cleaning DONE
- Filtering out isoforms DONE

	Predictor	Coverage
0	CADD	100.00
1	MetaRNN	100.00
2	DANN	99.89
3	BayesDel	99.88
4	GERP++	99.87
5	ClinPred	99.74
6	MetaSVM	99.67
7	MetaLR	99.67
8	PrimateAI	99.15
9	PROVEAN	98.46
10	DEOGEN2	98.30
11	FATHMM	98.20
12	LIST-S2	98.20
13	ESM1b	98.08
14	SIFT	97.85
15	gMVP	97.76

	Predictor	Coverage
15	gMVP	97.76
16	VARITY	97.30
17	MutationTaster	94.47
18	Eigen-PC-raw	93.49
19	Eigen-raw	93.49
20	MutationAssessor	91.40
21	M-CAP	89.37
22	LRT	89.35
23	REVEL	89.24
24	AlphaMissense	88.80
25	MPC	87.66
26	VEST4	87.30
27	PolyPhen	84.52
28	MutPred	61.56
29	EVE	53.76
30	MVP	27.10