Part_1_Extra_Credit

April 29, 2025

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Merge both dataframes vertically (same columns)
#merged_df = pd.concat([df_noncoding, df_coding], axis=0).reset_index(drop=True)

# Save to a new CSV file
merged_path = "Z.variantCall.SNPs_anno.complete.csv"
#merged_df.to_csv(merged_path, index=False)


merged_df = pd.read_csv(merged_path)
# Display basic info
merged_info = {
    "num_rows": merged_df.shape[0],
    "num_columns": merged_df.shape[1],
    "column_names_sample": merged_df.columns[:10].tolist()
}
merged_info
```

```
/var/folders/dh/ct60f64n7r30w2htbyy2py5c0000gn/T/ipykernel_28414/2326261280.py:1
0: DtypeWarning: Columns (19,20,21,35,36) have mixed types. Specify dtype option
on import or set low_memory=False.
  merged_df = pd.read_csv(merged_path)
```

[ ]:
```
{'num_rows': 3495851,
 'num_columns': 91,
 'column_names_sample': ['Gene',
  'Gene Full Name',
  'Chrom/Position',
  'Change',
  'Filter',
  'Mapping Quality (MQ)',
  'Genotype',
  'Frequency',
  'GQ Score',
  'Ref Depth']}
```

[2]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3495851 entries, 0 to 3495850
Data columns (total 91 columns):
 #   Column                          Dtype
---  ------                          -----
 0   Gene                            object
 1   Gene Full Name                  object
 2   Chrom/Position                  object
 3   Change                          object
 4   Filter                          object
 5   Mapping Quality (MQ)            float64
 6   Genotype                        object
 7   Frequency                       object
 8   GQ Score                        int64
 9   Ref Depth                       int64
 10  Alt Depth                       int64
 11  Top Consequence                 object
 12  Consequence                     object
 13  Impact                          object
 14  Splice Site                     object
 15  Max. Population Frequency       float64
 16  Max. Sub-Population Frequency   float64
 17  ClinVar Evaluation              object
 18  ClinVar Phenotype               object
 19  OMIM                            object
 20  HPO Phenotype                   object
 21  GenCC Disease                   object
 22  pLI                             object
 23  SIFT                            object
 24  PolyPhen                        object
 25  Effect Prediction               object
 26  Effect Score                    int64
 27  CADD Conservation               object
 28  Intolerance                     object
 29  GERP++ Score                    object
 30  PhyloP46 Conservation           object
 31  PhyloP46 Conservation Score     object
 32  HGVS (RefGene)                  object
 33  HGVS (VEP)                      object
 34  Gene Family                     object
 35  Gene Description                object
 36  GO Function                     object
 37  GO Cell Component               object
 38  GO Pathway                      object
 39  Literature                      object
 40  COSMIC Count                    object
```

```
41  COSMIC Types                    object
42  Regulatory                      object
43  Interpro                        object
44  Splicing dbscSNV11              object
45  SpliceAI Max. Score             object
46  SpliceAI Type (Max. Score)      object
47  SpliceAI Position (Max. Score)  object
48  M-CAP Pathogenicity Score       object
49  REVEL Pathogenicity Score       object
50  regSNP-intron Pathogenicity     object
51  Spidex Zscore                   object
52  Spidex Max Tissue               object
53  MANE SELECT                     object
54  MANE CLINICAL                   object
55  dbSNP                           object
56  Cytoband                        object
57  SegDup                          object
58  Yale MedExome                   float64
59  gnomAD Exome                    float64
60  gnomAD Genome                   float64
61  gnomADg (AFR)                   float64
62  gnomADg (AMI)                   float64
63  gnomADg (AMR)                   float64
64  gnomADg (ASJ)                   float64
65  gnomADg (EAS)                   float64
66  gnomADg (FIN)                   float64
67  gnomADg (MID)                   float64
68  gnomADg (NFE)                   float64
69  gnomADg (OTH)                   float64
70  gnomADg (SAS)                   float64
71  LOFTEE                          object
72  LOFTEE Details                  object
73  SIFT Prediction                 object
74  SIFT-4G Prediction              object
75  Polyphen2 HDIV Prediction       object
76  Polyphen2 HVAR Prediction       object
77  FATHMM Prediction               object
78  MetaSVM Prediction              object
79  LRT Prediction                  object
80  MutationTaster Prediction       object
81  MutationAssessor Prediction     object
82  MetaLR Prediction               object
83  PROVEAN Prediction              object
84  MVP Score                       object
85  MPC Score                       object
86  VEST4 Score                     object
87  PhyloP 46-way Score             object
88  PhyloP100 Vertebrate            object
```

```
 89  PhyloP30 Mammalian              object
 90  PrimateAI                       object
dtypes: float64(16), int64(4), object(71)
memory usage: 2.4+ GB
```

[3]:
```
merged_df
```

[3]:

	Gene	Gene Full Name	Chrom/Position	Change	Filter	\
0	NONE;DDX11L17	NaN	1:10247	SNP:T>C	PASS	
1	NONE;DDX11L17	NaN	1:10248	SNP:A>T	PASS	
2	DDX11L1;DDX11L17	NaN	1:12783	SNP:G>A	PASS	
3	WASH7P	NaN	1:14464	SNP:A>T	PASS	
4	WASH7P	NaN	1:15118	SNP:A>G	PASS	
…	…	…	…	…	…	
3495846	NONE;NONE	NaN	MT:8764	SNP:G>A	PASS	
3495847	NONE;NONE	NaN	MT:8860	SNP:A>G	PASS	
3495848	NONE;NONE	NaN	MT:14869	SNP:G>A	PASS	
3495849	NONE;NONE	NaN	MT:15326	SNP:A>G	PASS	
3495850	NONE;NONE	NaN	MT:15736	SNP:A>G	PASS	

	Mapping Quality (MQ)	Genotype	Frequency	GQ Score	Ref Depth	…	\
0	35.36	Hom_1/1	100%	6	0	…	
1	35.36	Hom_1/1	100%	6	0	…	
2	25.12	Het_0/1	70%	99	7	…	
3	33.27	Het_0/1	28%	99	26	…	
4	25.47	Het_0/1	42%	99	49	…	
…	…	…	…	…	…	…	
3495846	46.71	Hom_1/1	99%	99	1	…	
3495847	35.17	Hom_1/1	100%	99	0	…	
3495848	59.35	Hom_1/1	100%	99	0	…	
3495849	60.00	Hom_1/1	100%	99	0	…	
3495850	60.00	Hom_1/1	100%	99	0	…	

	MutationAssessor Prediction	MetaLR Prediction	PROVEAN Prediction	\
0	.	.	.	
1	.	.	.	
2	.	.	.	
3	.	.	.	
4	.	.	.	
…	…	…	…	
3495846	.	.	.	
3495847	.	.	.	
3495848	.	.	.	
3495849	.	.	.	
3495850	.	.	.	

	MVP Score	MPC Score	VEST4 Score	PhyloP 46-way Score	\

```
0                .          .            .                    .
1                .          .            .                    .
2                .          .            .                -0.783
3                .          .            .                -0.640
4                .          .            .                 0.621
…              …        …          …                  …
3495846          .          .            .                -0.903
3495847          .          .            .                -1.746
3495848          .          .            .                 4.578
3495849          .          .            .                -4.154
3495850          .          .            .                 3.798

         PhyloP100 Vertebrate PhyloP30 Mammalian PrimateAI
0                          .                   .         .
1                          .                   .         .
2                          .                   .         .
3                          .                   .         .
4                          .                   .         .
…                        …                 …       …
3495846                    .                   .         .
3495847                    .                   .         .
3495848                    .                   .         .
3495849                    .                   .         .
3495850                    .                   .         .

[3495851 rows x 91 columns]
```

[4]:
```
merged_df_chr22 = merged_df[merged_df["Chrom/Position"].str.startswith("22:")].
↪copy()
```

[5]:
```
merged_df_chr22
```

[5]:
```
                Gene   Gene Full Name Chrom/Position   Change Filter  \
3336481  NONE;DUXAP8              NaN    22:16050822  SNP:G>A   PASS
3336482  NONE;DUXAP8              NaN    22:16051249  SNP:T>C   PASS
3336483  NONE;DUXAP8              NaN    22:16051347  SNP:G>C   PASS
3336484  NONE;DUXAP8              NaN    22:16051453  SNP:A>C   PASS
3336485  NONE;DUXAP8              NaN    22:16051497  SNP:A>G   PASS
…              …              …            …      …    …
3495414         ARSA  arylsulfatase A    22:51064039  SNP:G>C   PASS
3495415         ARSA  arylsulfatase A    22:51064068  SNP:G>A   PASS
3495416         ARSA  arylsulfatase A    22:51064416  SNP:T>C   PASS
3495417         ARSA  arylsulfatase A    22:51065600  SNP:G>A   PASS
3495418          ACR          acrosin    22:51183255  SNP:A>G   PASS

         Mapping Quality (MQ) Genotype Frequency  GQ Score  Ref Depth  …  \
3336481                 29.34  Het_0/1       48%        99         17  …
```

```
3336482                58.93  Het_0/1       45%        99         21  …
3336483                36.08  Hom_1/1      100%        81          0  …
3336484                34.75  Het_0/1       44%        99         23  …
3336485                36.21  Hom_1/1      100%        99          0  …
…                      …      …       …      …        …  …
3495414                60.00  Hom_1/1      100%        77          0  …
3495415                60.00  Het_0/1       57%        99         10  …
3495416                60.00  Het_0/1       46%        99         20  …
3495417                60.00  Het_0/1       33%        99         16  …
3495418                40.28  Het_0/1       50%        99          5  …

         MutationAssessor Prediction MetaLR Prediction PROVEAN Prediction  \
3336481                            .                 .                   .
3336482                            .                 .                   .
3336483                            .                 .                   .
3336484                            .                 .                   .
3336485                            .                 .                   .
…                                  …                 …                   …
3495414                            .                 T                   N
3495415                            .                 .                   .
3496416                            .                 T                   N
3495417                            .                 .                   .
3495418                            M                 T                   N

        MVP Score MPC Score  VEST4 Score  PhyloP 46-way Score  \
3336481         .          .            .                -1.656
3336482         .          .            .                 0.058
3336483         .          .            .                 0.064
3336484         .          .            .                 0.058
3336485         .          .            .                 0.058
…           …        …           …                   …
3495414         .          .        0.031                -0.076
3495415         .          .            .                -1.434
3496416         .          .        0.149                 1.967
3495417         .          .            .                -1.003
3495418         .          .        0.053                 0.234

        PhyloP100 Vertebrate PhyloP30 Mammalian PrimateAI
3336481                    .                   .         .
3336482                    .                   .         .
3336483                    .                   .         .
3336484                    .                   .         .
3336485                    .                   .         .
…                          …                   …         …
3495414                0.122              -1.913         .
3495415                    .                   .         .
3496416                1.458               1.138         .
```

```
3495417                    .                    .          .
3495418               -0.457               -0.682          .

[45592 rows x 91 columns]
```

[6]:
```
df = merged_df
df = pd.read_csv("Z.variantCall.SNPs_anno.complete.csv")

# Filter for chromosome 22
df_chr22 = df[df["Chrom/Position"].str.startswith("22:")].copy()

# Split multiple gene entries
#df_chr22 = df_chr22.assign(Gene=df_chr22["Gene"].str.split(";")).
 ↪explode("Gene")

# Count number of mutations per gene

gene_mut_counts = df_chr22["Gene"].value_counts().reset_index()
gene_mut_counts.columns = ["Gene", "Mutation Count"]
```

```
/var/folders/dh/ct60f64n7r30w2htbyy2py5c0000gn/T/ipykernel_28414/3233792492.py:2
: DtypeWarning: Columns (19,20,21,35,36) have mixed types. Specify dtype option
on import or set low_memory=False.
  df = pd.read_csv("Z.variantCall.SNPs_anno.complete.csv")
```

[7]:
```
df_chr22['Top Consequence'].value_counts()
```

[7]:
```
intron_variant                          24731
intergenic_variant                      13520
upstream_gene_variant                    3802
non_coding_transcript_intron_variant      900
non_coding_transcript_exon_variant        840
3_prime_UTR_variant                       564
downstream_gene_variant                   368
missense_variant                          315
synonymous_variant                        308
5_prime_UTR_variant                        92
splice_polypyrimidine_tract_variant        56
splice_region_variant                      51
splice_donor_region_variant                10
splice_donor_variant                        8
splice_donor_5th_base_variant               7
splice_acceptor_variant                     6
stop_gained                                 5
start_lost                                  4
mature_miRNA_variant                        3
non_coding_transcript_splicing_variant      1
incomplete_terminal_codon_variant           1
```

```
Name: Top Consequence, dtype: int64
```

[16]:
```
df_chr22['Impact'].value_counts()
```

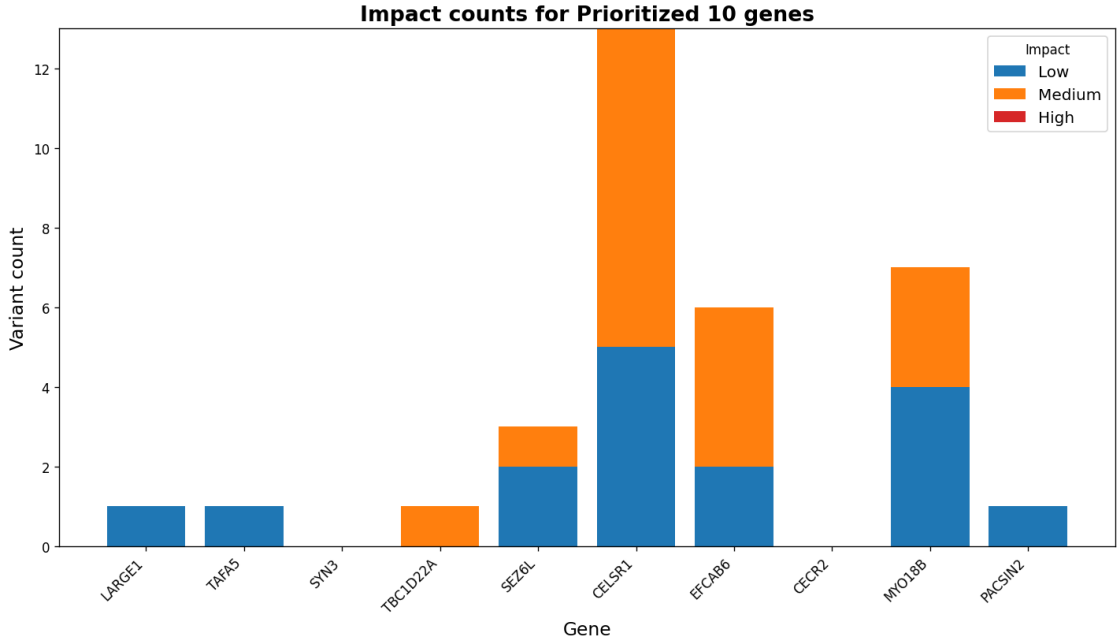
[16]:
```
.         44821
Low         433
Medium      315
High         23
Name: Impact, dtype: int64
```

[ ]:
```
# ----------------------------------------------------------------------
# 1. Keep only the genes of interest
# ----------------------------------------------------------------------
genes = [
    "LARGE1", "TAFA5", "SYN3", "TBC1D22A", "SEZ6L",
    "CELSR1", "EFCAB6", "CECR2", "MYO18B", "PACSIN2"
]
filtered = df_chr22[df_chr22["Gene"].isin(genes)]

# ----------------------------------------------------------------------
# 2. Sanity-check the raw mutation counts
# ----------------------------------------------------------------------
mutation_counts = (
    filtered["Gene"]
    .value_counts()
    .reindex(genes, fill_value=0)
)

print("Raw mutation counts:")
print(mutation_counts)


# ----------------------------------------------------------------------
# 3. Impact counts per gene (Low / Medium / High)
# ----------------------------------------------------------------------
impact_levels = ["Low", "Medium", "High"]

impact_counts = (
    filtered.groupby(["Gene", "Impact"])
            .size()
            .unstack(fill_value=0)            # Impact → columns
            .reindex(index=genes)             # keep our gene order
            .reindex(columns=impact_levels,   # ensure all 3 cols exist
                     fill_value=0)
)

print("\nImpact breakdown:")
```

```
print(impact_counts.head())

# ------------------------------------------------------------------
# 4. Stacked bar plot
# ------------------------------------------------------------------
plt.figure(figsize=(12, 7), dpi=120)

# three bars stacked for each gene
colors = ["#1f77b4", "#ff7f0e", "#d62728"]      # Low / Medium / High
bottom = None
for idx, impact in enumerate(["Low", "Medium", "High"]):
    plt.bar(
        impact_counts.index,
        impact_counts[impact],
        bottom=bottom,
        label=impact,
        color=colors[idx]
    )
    bottom = (
        impact_counts[impact] if bottom is None
        else bottom + impact_counts[impact]
    )

plt.title("Impact counts for Prioritized 10 genes",
          fontsize=16, weight="bold")
plt.xlabel("Gene", fontsize=14)
plt.ylabel("Variant count", fontsize=14)
plt.xticks(rotation=45, ha="right")
plt.legend(title="Impact", fontsize=12)
plt.tight_layout()

plt.savefig("impact_distribution_chr22_genes.png", bbox_inches="tight")
plt.show()
```

```
Raw mutation counts:
LARGE1      808
TAFA5       759
SYN3        641
TBC1D22A    550
SEZ6L       391
CELSR1      388
EFCAB6      349
CECR2       343
MYO18B      336
PACSIN2     330
Name: Gene, dtype: int64

Impact breakdown:
```

```
Impact     Low  Medium  High
Gene
LARGE1       1       0     0
TAFA5        1       0     0
SYN3         0       0     0
TBC1D22A     0       1     0
SEZ6L        2       1     0
```



[20]:
```
impact_levels = ["Low", "Medium", "High"]

impact_counts = (
    df_chr22[df_chr22["Gene"].isin(genes)]
      .groupby(["Gene", "Impact"]).size()
      .unstack(fill_value=0)                # Impact → columns
      .reindex(index=genes)                 # keep top-10 order
      .reindex(columns=impact_levels,       # ensure all three cols
               fill_value=0)
)

print("Impact breakdown (top-10 genes):")
print(impact_counts)
```

```
Impact breakdown (top-10 genes):
Impact     Low  Medium  High
Gene
LARGE1       1       0     0
TAFA5        1       0     0
```

```
SYN3        0       0     0
TBC1D22A    0       1     0
SEZ6L       2       1     0
CELSR1      5       8     0
EFCAB6      2       4     0
CECR2       0       0     0
MYO18B      4       3     0
PACSIN2     1       0     0
```

[22]:
```python
# ------------------------------------------------------------------
# 4. Top-Consequence breakdown for the same top-10 genes
# ------------------------------------------------------------------
# (a) overall frequency, just to eyeball the mix
tc_overall = (
    df_chr22[df_chr22["Gene"].isin(genes)]["Top Consequence"]
    .value_counts()
)
print("\nOverall Top Consequence counts (top-10 genes):")
print(tc_overall.head(15))      # show the 15 most common categories

# (b) per-gene matrix  (Gene × Top Consequence)
tc_per_gene = (
    df_chr22[df_chr22["Gene"].isin(genes)]
      .groupby(["Gene", "Top Consequence"]).size()
      .unstack(fill_value=0)
      .reindex(index=genes)          # keep descending-mutation order
)

print("\nTop Consequence breakdown per gene:")
print(tc_per_gene)

# Optional: save to CSV for further inspection
tc_per_gene.to_csv("chr22_top10_top_consequence_breakdown.csv")
```

```
Overall Top Consequence counts (top-10 genes):
intron_variant                 4801
3_prime_UTR_variant              31
upstream_gene_variant            19
missense_variant                 17
synonymous_variant               15
downstream_gene_variant           6
intergenic_variant                3
5_prime_UTR_variant               2
splice_donor_region_variant       1
Name: Top Consequence, dtype: int64

Top Consequence breakdown per gene:
```

Top Consequence	3_prime_UTR_variant	5_prime_UTR_variant
Gene		
LARGE1	0	1
TAFA5	0	1
SYN3	19	0
TBC1D22A	3	0
SEZ6L	4	0
CELSR1	0	0
EFCAB6	0	0
CECR2	5	0
MYO18B	0	0
PACSIN2	0	0

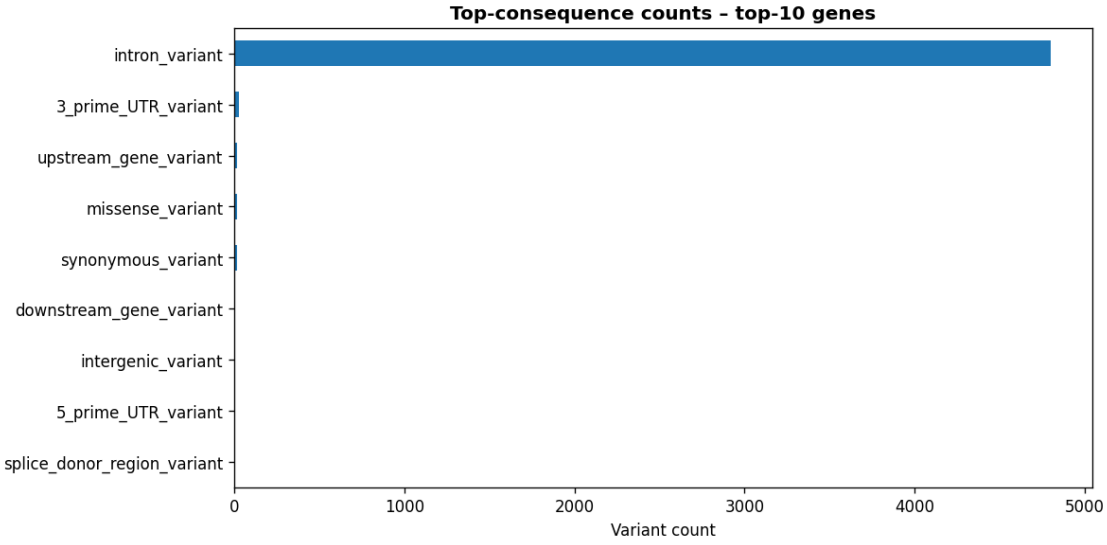
Top Consequence	downstream_gene_variant	intergenic_variant	intron_variant
Gene			
LARGE1	0	0	805
TAFA5	0	0	756
SYN3	0	3	618
TBC1D22A	5	0	540
SEZ6L	1	0	383
CELSR1	0	0	373
EFCAB6	0	0	343
CECR2	0	0	330
MYO18B	0	0	326
PACSIN2	0	0	327

Top Consequence	missense_variant	splice_donor_region_variant
Gene		
LARGE1	0	0
TAFA5	0	1
SYN3	0	0
TBC1D22A	1	0
SEZ6L	1	0
CELSR1	8	0
EFCAB6	4	0
CECR2	0	0
MYO18B	3	0
PACSIN2	0	0

Top Consequence	synonymous_variant	upstream_gene_variant
Gene		
LARGE1	1	1
TAFA5	0	1
SYN3	0	1
TBC1D22A	0	1
SEZ6L	2	0
CELSR1	5	2
EFCAB6	2	0

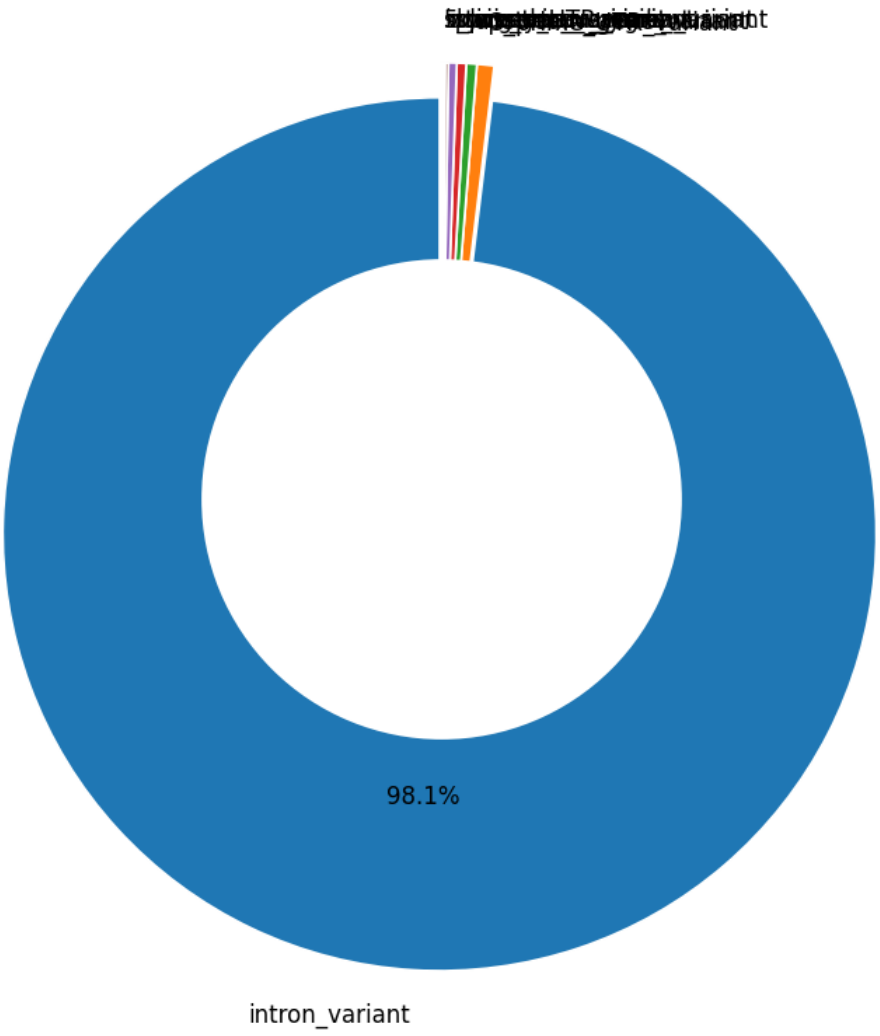
CECR2	0	8
MYO18B	4	3
PACSIN2	1	2

[23]:
```
plt.figure(figsize=(10, 5), dpi=120)
(
    tc_overall
    .sort_values(ascending=True)        # smallest on top
    .plot.barh(color="#1f77b4")
)

plt.title("Top-consequence counts - top-10 genes", weight="bold")
plt.xlabel("Variant count")
plt.tight_layout()
plt.savefig("tc_overall_bar.png", bbox_inches="tight")
plt.show()
```



[24]:
```
explode = [0.08 if idx==0 else 0        # explode only first slice (intron)
           for idx in range(len(tc_overall))]

plt.figure(figsize=(7, 7), dpi=120)
plt.pie(tc_overall,
        labels=tc_overall.index,
        explode=explode,
        autopct=lambda p: f"{p:.1f}%" if p > 1 else "",    # hide <1 %
        startangle=90,
        wedgeprops=dict(edgecolor="white"))
centre_circ = plt.Circle((0, 0), 0.55, fc="white")        # donut hole
plt.gca().add_artist(centre_circ)
```

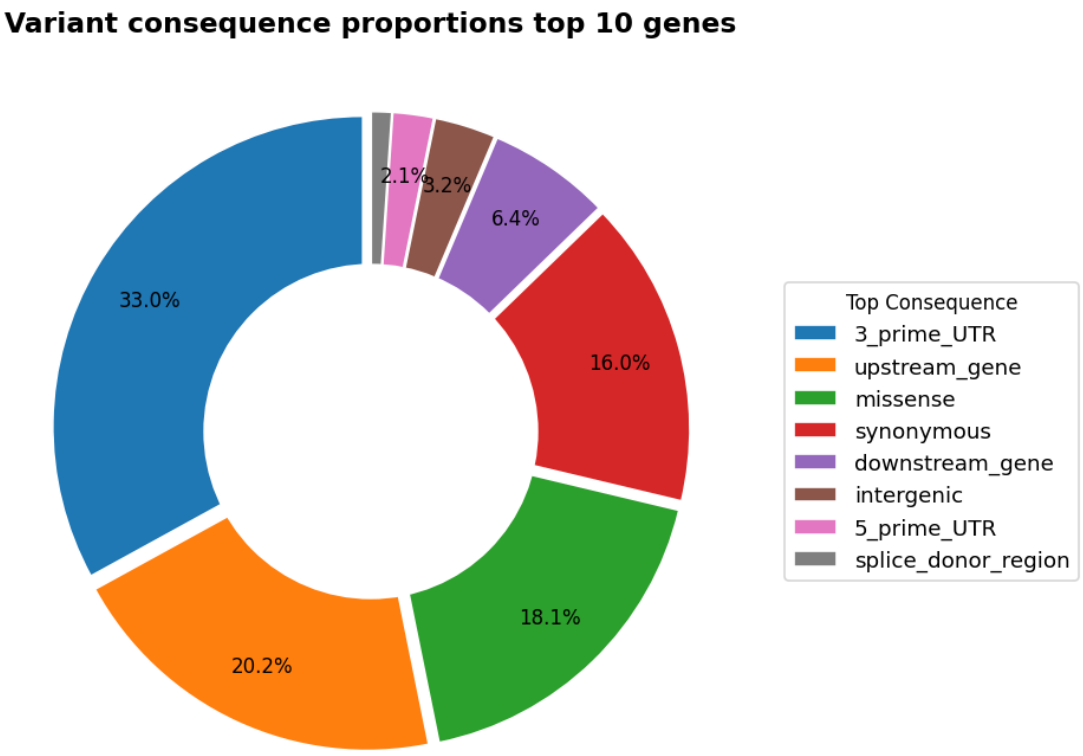
```
plt.title("Variant consequence proportions - top-10 genes",
          weight="bold")
plt.tight_layout()
plt.savefig("tc_overall_donut.png", bbox_inches="tight")
plt.show()
```



[27]:
```
tc_no_intron = tc_overall.drop("intron_variant")

fig, ax = plt.subplots(figsize=(7, 7), dpi=120)
ax.pie(tc_no_intron,
       labels=tc_no_intron.index,
```

```
        autopct=lambda p: f"{p:.1f}%" if p > 2 else "",  # hide <2 %
        startangle=90,
        wedgeprops=dict(edgecolor="white"))
ax.add_artist(plt.Circle((0, 0), 0.55, fc="white"))
ax.set_title("Variant consequence proportions - top-10 genes (EXCL. intron)",
             weight="bold")
plt.tight_layout()
plt.savefig("tc_overall_donut_excl_intron.png", bbox_inches="tight")
plt.close()
```

[ ]:
```
# tc_no_intron already defined  tc_overall.drop("intron_variant")
labels = tc_no_intron.index.str.replace("_variant", "", regex=False)  # shorter
sizes  = tc_no_intron.values
explode = [0.03] * len(sizes)        # uniform gap

fig, ax = plt.subplots(figsize=(8, 6), dpi=120)

wedges, texts, autotexts = ax.pie(
    sizes,
    explode=explode,
    startangle=90,
    labels=None,                     # no direct labels
    autopct=lambda p: f"{p:.1f}%" if p >= 2 else "",
    pctdistance=0.80,
    wedgeprops=dict(edgecolor="white", linewidth=1)
)

# Donut hole
ax.add_artist(plt.Circle((0, 0), 0.54, fc="white"))

# Legend on the right
ax.legend(wedges,
          labels,
          title="Top Consequence",
          loc="center left",
          bbox_to_anchor=(1.02, 0.5),
          fontsize=11)

ax.set_title("Variant consequence proportions top 10 genes",
             fontsize=14, weight="bold")
plt.tight_layout()
plt.savefig("tc_overall_donut_excl_intron_clean.png", bbox_inches="tight")
plt.show()
```



[ ]:
```
df_chr22['Polyphen2 HVAR Prediction']
```

[ ]:
```
3336481    .
3336482    .
3336483    .
3336484    .
3336485    .
          ..
3495414    T
3495415    .
3495416    T
3495417    .
3495418    T
Name: SIFT4G, Length: 45592, dtype: object
```

[57]:
```
df = merged_df
df = pd.read_csv("Z.variantCall.SNPs_anno.coding.csv")

# Filter for chromosome 22
df_chr22 = df[df["Chrom/Position"].str.startswith("22:")].copy()
```

[58]:
```
# exact column names in your frame
sift_col  = "SIFT-4G Prediction"
poly_col  = "Polyphen2 HVAR Prediction"    # ← note the spacing/case

# numeric-ise CADD (mixed floats / strings → float + NaN)
df_chr22["CADD_Conservation"] = pd.to_numeric(
    df_chr22["CADD Conservation"], errors="coerce"
)

# normalise the prediction symbols
for col in [sift_col, poly_col]:
    df_chr22[col] = df_chr22[col].fillna(".").astype(str).str.strip()

# non-synonymous list
non_syn = {
    "missense_variant", "stop_gained", "start_lost",
    "splice_donor_variant", "splice_acceptor_variant",
    "splice_region_variant", "stop_lost",
    "frameshift_variant", "inframe_insertion", "inframe_deletion"
}

# four-way mask
mask = (
    df_chr22["Top Consequence"].isin(non_syn) &
    (df_chr22["CADD_Conservation"] > 15) &
    df_chr22[sift_col].isin({"D", "."}) &
    df_chr22[poly_col].isin({"D", "P", "."})
)

df_chr22_filt = df_chr22.loc[mask].copy()

print(f"Before: {len(df_chr22):,}")
print(f"After : {len(df_chr22_filt):,} (passed all filters)")
```

```
Before: 798
After : 21 (passed all filters)
```

[59]:
```
df_chr22_filt["Gene"].value_counts()
```

[59]:
```
HPS4        2
GAB4        1
APOL5       1
SMC1B       1
EFCAB6      1
TTLL12      1
APOBEC3H    1
C1QTNF6     1
CIMIP4      1
```

```
APOL4       1
PRR14L      1
CLDN5       1
PLA2G3      1
NEFH        1
RFPL1       1
CCDC116     1
GGT2P       1
RIMBP3      1
TBX1        1
PRR34       1
Name: Gene, dtype: int64
```

[65]:
```
df_chr22_filt['Impact']
```

[65]:
```
30360      High
30400      High
30402    Medium
30421    Medium
30464    Medium
30477    Medium
30614    Medium
30615    Medium
30643    Medium
30646    Medium
30680    Medium
30697    Medium
30736    Medium
30744    Medium
30775    Medium
30786    Medium
30841    Medium
30941    Medium
30951    Medium
30986    Medium
31004    Medium
Name: Impact, dtype: object
```