SV-eQTL analysis

Imports and Consts

```
import pandas as pd
import numpy as np
import statsmodels.api
import scipy.stats
import seaborn as sns
import matplotlib.pyplot as plt
```

```
DATASET_PAH = "./Dataset/"
RNAFILENAME = "GD660.GeneQuantRPKM.txt"
VCFFILENAME = "ALL.wgs.mergedSV.v8.20130502.svs.genotypes.vcf"
CHOSEN_CHROMOSOME = 19
```

Load and prepare data

```
rnaSeqData = pd.read_csv(DATASET_PAH + RNAFILENAME, delimiter="\t")
```

```
rnaSeqData
```

	TargetID	Gene_Symbol	Chr	Coord	HG00096.1.M_111124_6	HG00097.7.M_120219_2
0	ENSG00000225538.1	ENSG00000225538.1	11	55850277	0.00000	0.00000
1	ENSG00000237851.1	ENSG00000237851.1	6	143109260	0.00000	0.00000
2	ENSG00000243765.1	ENSG00000243765.1	15	58442766	0.00000	0.00000
3	ENSG00000257527.1	ENSG00000257527.1	16	18505708	0.70561	0.66697
4	ENSG00000212855.5	ENSG00000212855.5	Y	9578193	0.00000	0.00000
...	...	...	...	...	...	...
53929	ENSG00000172297.6	ENSG00000172297.6	Y	27600708	0.13907	0.10224
53930	ENSG00000259738.1	ENSG00000259738.1	15	59157205	0.00000	0.13191
53931	ENSG00000212040.1	ENSG00000212040.1	14	101498324	0.00000	0.00000
53932	ENSG00000125266.5	ENSG00000125266.5	13	107187462	0.12923	0.07601
53933	ENSG00000230711.2	ENSG00000230711.2	6	168690030	0.20363	0.18754

53934 rows × 664 columns

```
f = open(DATASET_PAH + VCFFILENAME, "r")
columns = None
for l in f:
    if "#CHROM" in l:
        columns = l.split("\t")
        break
```

```
columns[0] = columns[0].replace("#", "")
```

```
columns[len(columns) - 1] = columns[len(columns) - 1].replace("\n", "")
```

```
vcfSourceFile = pd.read_csv(
    DATASET_PAH + VCFFILENAME,
    sep="\t",
    comment="#",
    names=columns,
    header=None,
)
```

```
/tmp/ipykernel_189579/242614598.py:1: DtypeWarning: Columns (0) have mixed types. Specify dtype option
on import or set low_memory=False.
  vcfSourceFile = pd.read_csv(
```

There are groups of values in 'ALT' column that need to be split

```
vcfSourceFile[vcfSourceFile["CHROM"] == CHOSEN_CHROMOSOME]["ALT"].unique()
```

```
array(['<CN2>', '<CN0>,<CN2>', '<CN0>', 'G', 'A', 'C', '<INS:ME:ALU>',
       '<INS:ME:SVA>', 'T', '<CN0>,<CN2>,<CN3>', '<INV>', '<INS:MT>',
       '<CN2>,<CN3>', '<INS:ME:LINE1>'], dtype=object)
```

```
def processChr(row):
    if row["Chr"] not in ["X", "Y", "M"]:
        row["Chr"] = str(row["Chr"])
    return row


rnaSeqData = rnaSeqData.apply(lambda x: processChr(x), axis=1)
```

```
rnaSeqData["Chr"].unique()
```

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

```
def processCHROM(row):
    if row["CHROM"] not in ["X", "Y", "M"]:
        row["CHROM"] = str(row["CHROM"])
    return row


vcfSourceFile = vcfSourceFile.apply(lambda x: processCHROM(x), axis=1)

vcfSourceFile["CHROM"].unique()
```

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

Investigating the number of datapoints for each chromosome to choose one with reasonable size

```
vcfSourceFile.groupby(by="CHROM").size()
```

```
CHROM
1     4671
10    3126
11    3375
12    3299
13    2485
14    2097
15    1867
16    2062
17    1926
18    2005
19    1621
2     5642
20    1569
21     877
22     848
3     4811
4     4780
5     4425
6     4187
7     4200
8     3681
9     3001
X     2263
dtype: int64
```

```
rnaSeqData.groupby(by="Chr").size()
```

```
Chr
1     5172
10    2199
11    3121
12    2747
13    1185
14    2182
15    2021
16    2292
17    2207
18     562
19    1939
2     3872
20    1276
21     694
22    1187
3     2917
4     2494
5     2734
6     2794
7     2770
8     2315
9     2334
M       37
X     2328
Y      555
dtype: int64
```

```
vcfSourceFile.columns
```

```
Index(['CHROM', 'POS', 'ID', 'REF', 'ALT', 'QUAL', 'FILTER', 'INFO', 'FORMAT',
       'HG00096',
       ...
       'NA21128', 'NA21129', 'NA21130', 'NA21133', 'NA21135', 'NA21137',
       'NA21141', 'NA21142', 'NA21143', 'NA21144'],
      dtype='object', length=2513)
```

```
vcfSourceFile["CHROM"] = vcfSourceFile["CHROM"].apply(lambda x: str(x))
# I pick only 1 chromosome to make sure the computation time is reasonable
vcfSourceFile = vcfSourceFile[vcfSourceFile["CHROM"] == str(CHOSEN_CHROMOSOME)].copy()
```

```
vcfSourceFile
```

	CHROM	POS	ID	REF	ALT	QUAL	FILTER	
61640	19	251548	DUP_gs_CNV_19_251548_283564	C	<CN2>	.	PASS	AC=2;AF=0.0003993
61641	19	266428	DUP_uwash_chr19_266428_341459	T	<CN2>	.	PASS	AC=4;AF=0.0007987
61642	19	286841	DUP_gs_CNV_19_286841_307454	T	<CN0>, <CN2>	.	PASS	AC=3,3;AF=0.00
61643	19	293937	BI_GS_DEL1_B2_P2673_28	G	<CN0>	100	PASS	AC=8;AF=0.001597
61644	19	323368	DUP_gs_CNV_19_323368_334484	A	<CN0>, <CN2>	.	PASS	AC=1,8;AF=0.00
...	...	...	...	...	...	...	...	
63256	19	58999008	SVA_umary_SVA_769	T	<INS:ME:SVA>	.	.	AC=6;AF=0.0011980
63257	19	59015810	BI_GS_DEL1_B2_P2732_9	G	<CN0>	100	PASS	AC=2;AF=0.0003993
63258	19	59020256	UW_VH_3019	C	<CN0>	100	PASS	AC=1;AF=0.0001996
63259	19	59021221	BI_GS_DEL1_B5_P2732_16	T	<CN0>	100	PASS	AC=1;AF=0.0001996
63260	19	59035015	YL_CN_JPT_4322	C	<CN0>	100	PASS	AC=2;AF=0.0003993

1621 rows × 2513 columns

```
newColumns = []
for i in range(len(rnaSeqData.columns)):
    newColumns.append(rnaSeqData.columns[i].split(".")[0])
```

```
rnaSeqData.columns = newColumns
```

```
rnaSeqData = rnaSeqData[rnaSeqData["Chr"] == str(CHOSEN_CHROMOSOME)].copy()
```

```
rnaSeqData = rnaSeqData.loc[:, ~rnaSeqData.columns.duplicated()].copy()
```

```
commonColumns = rnaSeqData.columns.intersection(vcfSourceFile.columns)
```

```
commonColumns
```

```
Index(['HG00096', 'HG00097', 'HG00099', 'HG00100', 'HG00101', 'HG00102',
       'HG00103', 'HG00105', 'HG00106', 'HG00108',
       ...
       'NA20809', 'NA20810', 'NA20811', 'NA20812', 'NA20813', 'NA20814',
       'NA20815', 'NA20819', 'NA20826', 'NA20828'],
      dtype='object', length=445)
```

```
vcfSourceFile["QUAL"] = vcfSourceFile["QUAL"].apply(lambda x: int(x) if x != "." else 0)
# vcfSourceFile = vcfSourceFile[vcfSourceFile["QUAL"] > 90]
# Filtering by quality significantly reduces the amount of unique SVs, so I refrain from doing so
```

```
vcfSourceFile["ALT"] = vcfSourceFile["ALT"].apply(lambda x: x.split(","))
```

```
import warnings
```

```
warnings.filterwarnings("ignore")
```

The aim is to investigate the effect of each SV separately, so for each element in the ALT column I prepare the appropriate X and y vectors assuming that 0|0 -> 0, x|0 -> 1, 0|x -> 1, x|x -> 2

```
def processRow(row, allRows):
    rows = []
    for i in range(len(row["ALT"])):
        # for each element split each common column by | and if any value in ALT is bigger than i+1 set
        rows.append(row.copy())
        rows[i]["ALT"] = rows[i]["ALT"][i]
        for column in commonColumns:
            vals = row[column].split("|")
            if not (int(vals[0]) == 0 or int(vals[0]) == i + 1) or not (
                int(vals[1]) == 0 or int(vals[1]) == i + 1
            ):
                rows[i][column] = -1
            else:
                rows[i][column] = sum(np.array(vals, dtype=np.int64)) / (i + 1)
        allRows.append(rows[i].to_numpy())
```

```
allRows = []
_ = vcfSourceFile.apply(lambda x: processRow(x, allRows), axis=1)
```

```
vcfSourceFile = pd.DataFrame(allRows, columns=vcfSourceFile.columns)
```

```
for column in commonColumns:
    vcfSourceFile[column] = vcfSourceFile[column].apply(lambda x: int(x))
```

```
vcfSourceFile
```

	CHROM	POS	ID	REF	ALT	QUAL	FILTER	
0	19	251548	DUP_gs_CNV_19_251548_283564	C	<CN2>	0	PASS	AC=2;AF=0.00039936
1	19	266428	DUP_uwash_chr19_266428_341459	T	<CN2>	0	PASS	AC=4;AF=0.00079872
2	19	286841	DUP_gs_CNV_19_286841_307454	T	<CN0>	0	PASS	AC=3,3;AF=0.000
3	19	286841	DUP_gs_CNV_19_286841_307454	T	<CN2>	0	PASS	AC=3,3;AF=0.000
4	19	293937	BI_GS_DEL1_B2_P2673_28	G	<CN0>	100	PASS	AC=8;AF=0.0015974
...	...	...	...	...	...	...	...	
1727	19	58999008	SVA_umary_SVA_769	T	<INS:ME:SVA>	0	.	AC=6;AF=0.0011980
1728	19	59015810	BI_GS_DEL1_B2_P2732_9	G	<CN0>	100	PASS	AC=2;AF=0.00039936
1729	19	59020256	UW_VH_3019	C	<CN0>	100	PASS	AC=1;AF=0.00019968
1730	19	59021221	BI_GS_DEL1_B5_P2732_16	T	<CN0>	100	PASS	AC=1;AF=0.00019968
1731	19	59035015	YL_CN_JPT_4322	C	<CN0>	100	PASS	AC=2;AF=0.00039936

1732 rows × 2513 columns

```
rnaSeqData = rnaSeqData.reset_index()
```

Finding 3 closest RNA elements for each gene

```
def findClosestRna(rnaRow, closestRna, vcfRow):
    distance = np.abs(vcfRow["POS"] - rnaRow["Coord"])
```

```
    closestRna.append([distance, rnaRow.name])


def mergeRnaOnClosest(row, rna):
    closestRna = []
    rna.apply(lambda x: findClosestRna(x, closestRna, row), axis=1)
    # Merge on closestRna
    # print(closestRna["closest"])

    closestRnaDf = pd.DataFrame(closestRna, columns=["Distance", "Name"])

    row["RNAIndex"] = (
        closestRnaDf.sort_values(by="Distance", ascending=True)
        .head(3)["Name"]
        .to_list()
    )
    return row


vcfSourceFile = vcfSourceFile.reset_index()
vcfRnaMerged = vcfSourceFile.apply(
    lambda x: mergeRnaOnClosest(x, rnaSeqData), axis=1
).reset_index()
```

```
vcfRnaMerged = vcfRnaMerged.explode("RNAIndex")
```

```
vcfRnaMerged["RNAIndex"]
```

```
0       1829
0        691
0       1456
1        691
1       1829
        ...
1730    1900
1730    1045
1731    1900
1731     457
1731    1122
Name: RNAIndex, Length: 5196, dtype: object
```

```
rnaBaseColumns = ["TargetID", "Gene_Symbol", "Chr", "Coord"]
vcfBaseColumns = [
    "CHROM",
    "POS",
    "ID",
    "REF",
    "ALT",
    "QUAL",
    "FILTER",
    "INFO",
    "FORMAT",
]
```

```
vcfRnaMerged["ALT"].unique()
```

```
array(['<CN2>', '<CN0>', 'G', 'A', 'C', '<INS:ME:ALU>', '<INS:ME:SVA>',
       'T', '<CN3>', '<INV>', '<INS:MT>', '<INS:ME:LINE1>'], dtype=object)
```

```
vcfSourceFile[vcfBaseColumns + list(commonColumns)][
    vcfSourceFile["CHROM"] == str(CHOSEN_CHROMOSOME)
]
```

	CHROM	POS	ID	REF	ALT	QUAL	FILTER	
0	19	251548	DUP_gs_CNV_19_251548_283564	C	<CN2>	0	PASS	AC=2;AF=0.00039936
1	19	266428	DUP_uwash_chr19_266428_341459	T	<CN2>	0	PASS	AC=4;AF=0.00079872
2	19	286841	DUP_gs_CNV_19_286841_307454	T	<CN0>	0	PASS	AC=3,3;AF=0.000
3	19	286841	DUP_gs_CNV_19_286841_307454	T	<CN2>	0	PASS	AC=3,3;AF=0.000
4	19	293937	BI_GS_DEL1_B2_P2673_28	G	<CN0>	100	PASS	AC=8;AF=0.0015974
...	...	...	...	...	...	...	...	
1727	19	58999008	SVA_umary_SVA_769	T	<INS:ME:SVA>	0	.	AC=6;AF=0.0011980
1728	19	59015810	BI_GS_DEL1_B2_P2732_9	G	<CN0>	100	PASS	AC=2;AF=0.00039936
1729	19	59020256	UW_VH_3019	C	<CN0>	100	PASS	AC=1;AF=0.00019968
1730	19	59021221	BI_GS_DEL1_B5_P2732_16	T	<CN0>	100	PASS	AC=1;AF=0.00019968
1731	19	59035015	YL_CN_JPT_4322	C	<CN0>	100	PASS	AC=2;AF=0.00039936

1732 rows × 454 columns

```
rnaSeqData[rnaBaseColumns + list(commonColumns)][
    rnaSeqData["Chr"] == str(CHOSEN_CHROMOSOME)
]
```

	TargetID	Gene_Symbol	Chr	Coord	HG00096	HG00097	HG00099	HG00100	HG00101
0	ENSG00000243642.1	ENSG00000243642.1	19	58581338	0.00000	0.00000	0.00000	0.00000	0.00000
1	ENSG00000221504.1	ENSG00000221504.1	19	32417479	0.00000	0.00000	0.00000	0.00000	0.00000
2	ENSG00000105357.9	ENSG00000105357.9	19	50706885	0.00000	0.00893	0.01107	0.00000	0.00000
3	ENSG00000131037.8	ENSG00000131037.8	19	55583407	0.30019	0.36200	0.43385	0.35353	0.24540
4	ENSG00000007255.5	ENSG00000007255.5	19	45681485	20.41195	19.88715	19.50962	13.88066	13.49203
...	...	...	...	...	...	...	...	...	...
1934	ENSG00000213996.3	ENSG00000213996.3	19	19384074	0.00000	0.00000	0.00000	0.00000	0.00000
1935	ENSG00000244253.1	ENSG00000244253.1	19	40079595	0.00000	0.00000	0.00000	0.00000	0.00000
1936	ENSG00000130309.4	ENSG00000130309.4	19	17666460	11.26085	10.45836	13.02215	11.28219	13.15124
1937	ENSG00000083842.6	ENSG00000083842.6	19	58790318	2.14528	2.32057	2.45383	1.55192	2.37085
1938	ENSG00000182310.7	ENSG00000182310.7	19	52196593	0.67427	0.45852	0.22117	0.13868	0.37363

1939 rows × 449 columns

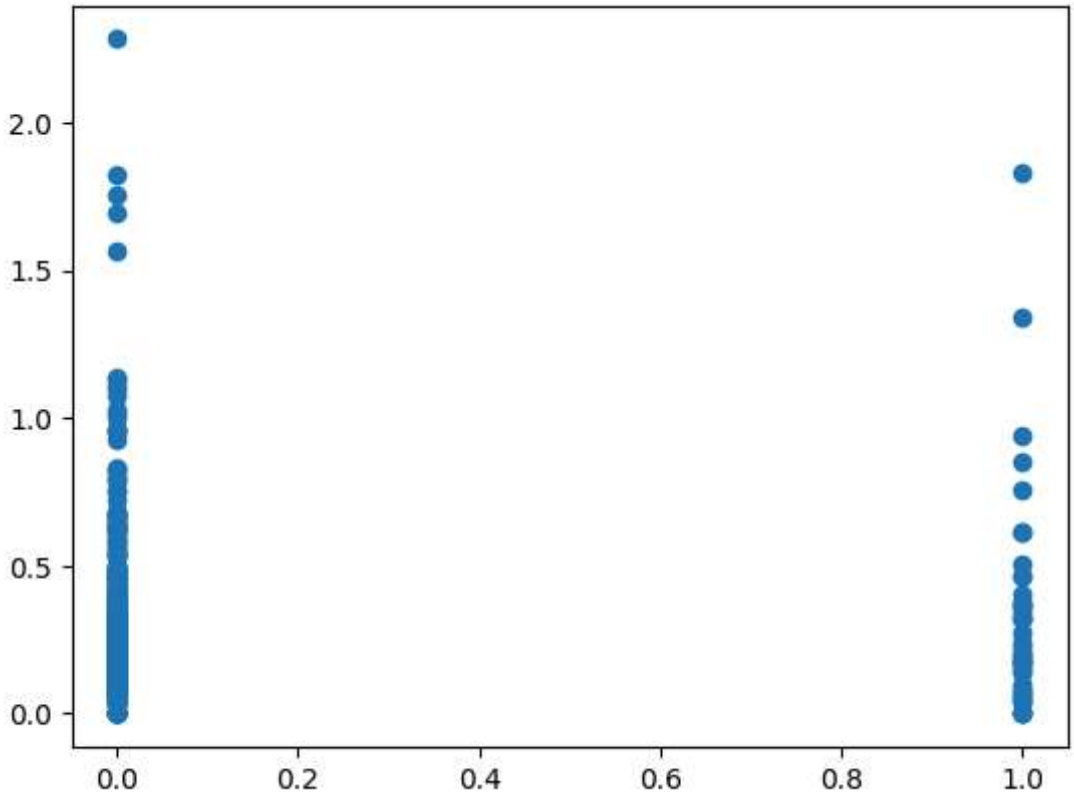
Perform linear regression

```
vcfRnaMerged.iloc[400]["ALT"]
```

```
'<INS:ME:ALU>'
```

```
plt.scatter(
    x=vcfRnaMerged.iloc[400][commonColumns].astype(int).to_frame().T.to_numpy()[0],
    y=rnaSeqData.iloc[vcfRnaMerged.iloc[400]["RNAIndex"]][commonColumns],
)
plt.show()

# The goal is to fit a line to such data, in some cases there will be also "2" x
```




```
p_values = {}
for index, row in vcfRnaMerged.iterrows():
    numpy_data = np.array(
        [
            row[commonColumns].astype(int).to_frame().T.to_numpy()[0],
            rnaSeqData.iloc[row["RNAIndex"]][commonColumns],
        ]
    )

    df = pd.DataFrame(data=numpy_data).T
    df.columns = ["x", "y"]
    df = df[df["x"] != -1]
    mod = statsmodels.api.OLS(
        np.asarray(df["y"], dtype=np.float64),
        df["x"].astype(float).to_frame().T.to_numpy()[0],
    )
    fii = mod.fit()

    p_values[row.name] = fii.summary2().tables[1]["P>|t|"]
```

```
p_values = {k: v.to_numpy()[0] for k, v in p_values.items() if v.to_numpy()[0] <= 0.05}
```

```
d = {}
for key in p_values.keys():
    d[key] = {}
    d[key]["GENEPOS"] = vcfRnaMerged.iloc[key]["POS"]
    d[key]["GENEID"] = vcfRnaMerged.iloc[key]["ID"]
    d[key]["ALT"] = vcfRnaMerged.iloc[key]["ALT"]
    d[key]["RNAID"] = rnaSeqData.iloc[vcfRnaMerged.iloc[key]["RNAIndex"]]["TargetID"]
    d[key]["RNACOORD"] = rnaSeqData.iloc[vcfRnaMerged.iloc[key]["RNAIndex"]]["Coord"]
    d[key]["SIG"] = p_values[key]
```

```
df = pd.DataFrame(d)
```

As a result 223 SV-eQTL have been discovered

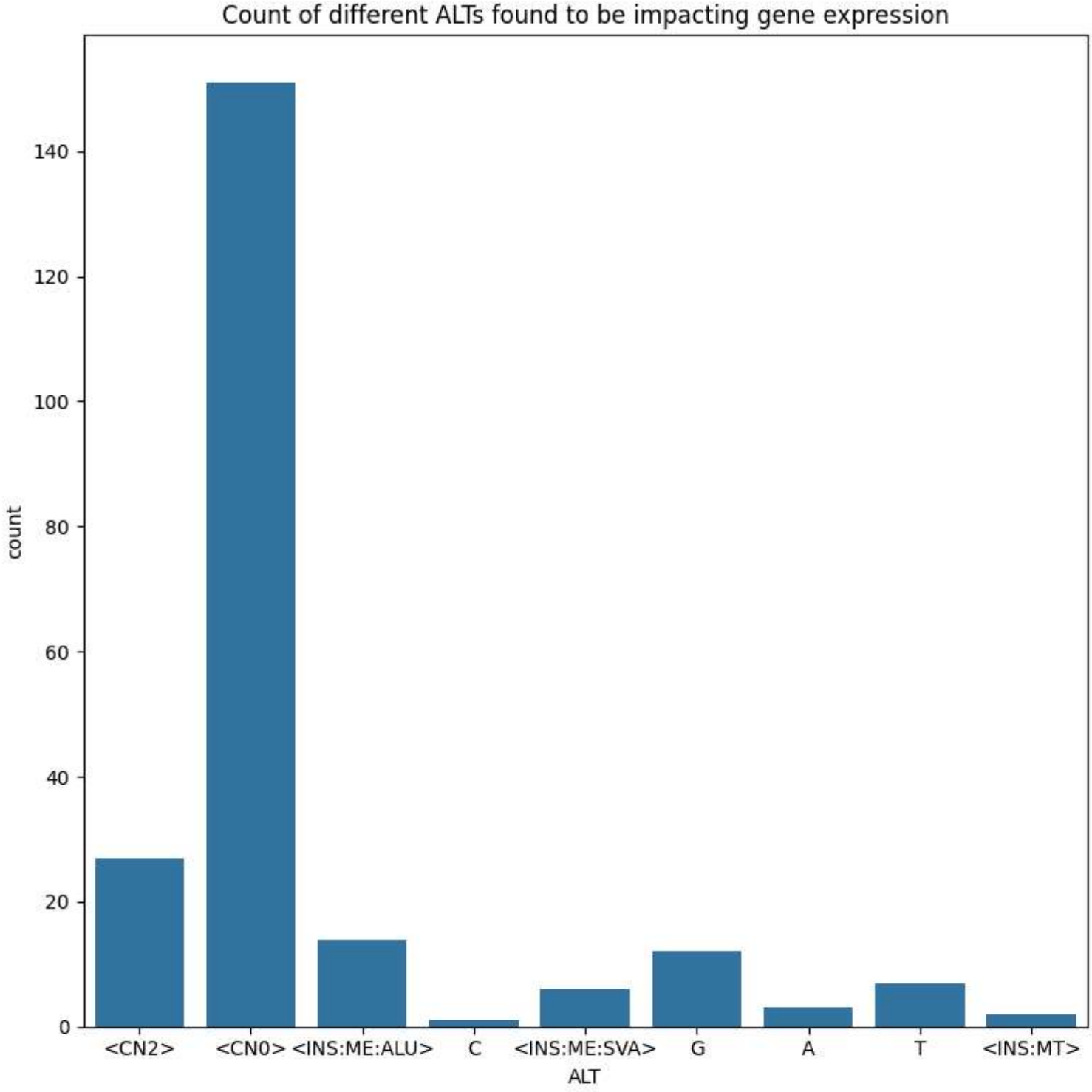
```
pd.set_option("display.float_format", "{:.2g}".format)
df.T
```

	GENEPOS	GENEID	ALT	RNAID	RNACOORD	SIG
10	286841	DUP_gs_CNV_19_286841_307454	<CN2>	ENSG00000105556.3	344791	1.1e-99
13	293937	BI_GS_DEL1_B2_P2673_28	<CN0>	ENSG00000105556.3	344791	4e-22
14	293937	BI_GS_DEL1_B2_P2673_28	<CN0>	ENSG00000222329.1	223261	4e-22
18	323368	DUP_gs_CNV_19_323368_334484	<CN2>	ENSG00000105556.3	344791	0.036
19	323368	DUP_gs_CNV_19_323368_334484	<CN2>	ENSG00000141934.3	291435	0.0062
...	...	...	...	...	...	...
1655	19432449	BI_GS_DEL1_B2_P2692_279	<CN0>	ENSG00000213996.3	19384074	1.9e-05
1657	19436758	YL_CN_GWD_5232	<CN0>	ENSG00000105705.8	19431318	2.7e-06
1660	19454338	ALU_umary_ALU_11889	<INS:ME:ALU>	ENSG00000105705.8	19431318	0.00012
1692	19985453	DEL_pindel_48427	G	ENSG00000256771.1	19976714	0.0037
1693	19985453	DEL_pindel_48427	G	ENSG00000184635.7	20011722	7.2e-19

223 rows × 6 columns

```
df.T.to_csv("regressionResults.csv")
```

```
plt.figure(figsize=(8, 8))
plt.title("Count of different ALTs found to be impacting gene expression")
sns.countplot(x="ALT", data=df.T)
plt.tight_layout()
plt.show()
```


Count of different ALTs found to be impacting gene expression
count
0
20
40
60
80
100
120
140
<CN2>
<CN0>
<INS:ME:ALU>
C
<INS:ME:SVA>
G
A
T
<INS:MT>
ALT