QUANTITATIVE TRAIT LOCI are regions of the genome associated with traits such as height, BMI etc.. If the trait is an expression of a gene then we are faced with an eQTL.

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/"
```

Load and preprocess data

```
rnaSeqData = pd.read_csv(DATASET_PAH + "GD660.GeneQuantRPKM.txt", 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 + "ALL.wgs.mergedSV.v8.20130502.svs.genotypes.vcf", "r")
columns = None
for l in f:
    if "#CHROM" in l:
        columns = l.split("\t")
        break
```

```
columns
```

```
['#CHROM',
 'POS',
 'ID',
 'REF',
 'ALT',
 'QUAL',
 'FILTER',
 'INFO',
 'FORMAT',
 'HG00096',
 'HG00097',
 'HG00099',
 'HG00100',
 'HG00101',
 'HG00102',
 'HG00103',
 'HG00105',
 'HG00106',
 'HG00107',
 'HG00108',
 'HG00109',
 'HG00110',
 'HG00111',
 'HG00112',
 'HG00113',
 'HG00114',
 'HG00115',
 'HG00116',
 'HG00117',
 'HG00118',
 'HG00119',
 'HG00120',
 'HG00121',
 'HG00122',
 'HG00123',
 'HG00125',
 'HG00126',
 'HG00127',
 'HG00128',
 'HG00129',
 'HG00130',
 'HG00131',
 'HG00132',
 'HG00133',
 'HG00136',
 'HG00137',
 'HG00138',
 'HG00139',
 'HG00140',
 'HG00141',
 'HG00142',
 'HG00143',
 'HG00145',
 'HG00146',
 'HG00148',
 'HG00149',
 'HG00150',
 'HG00151',
 'HG00154',
 'HG00155',
 'HG00157',
 'HG00158',
 'HG00159',
 'HG00160',
 'HG00171',
```

'HG00173',
'HG00174',
'HG00176',
'HG00177',
'HG00178',
'HG00179',
'HG00180',
'HG00181',
'HG00182',
'HG00183',
'HG00185',
'HG00186',
'HG00187',
'HG00188',
'HG00189',
'HG00190',
'HG00231',
'HG00232',
'HG00233',
'HG00234',
'HG00235',
'HG00236',
'HG00237',
'HG00238',
'HG00239',
'HG00240',
'HG00242',
'HG00243',
'HG00244',
'HG00245',
'HG00246',
'HG00250',
'HG00251',
'HG00252',
'HG00253',
'HG00254',
'HG00255',
'HG00256',
'HG00257',
'HG00258',
'HG00259',
'HG00260',
'HG00261',
'HG00262',
'HG00263',
'HG00264',
'HG00265',
'HG00266',
'HG00267',
'HG00268',
'HG00269',
'HG00271',
'HG00272',
'HG00273',
'HG00274',
'HG00275',
'HG00276',
'HG00277',
'HG00278',
'HG00280',
'HG00281',
'HG00282',
'HG00284',
'HG00285',
'HG00288',
'HG00290',

'HG00304',
'HG00306',
'HG00308',
'HG00309',
'HG00310',
'HG00311',
'HG00313',
'HG00315',
'HG00318',
'HG00319',
'HG00320',
'HG00321',
'HG00323',
'HG00324',
'HG00325',
'HG00326',
'HG00327',
'HG00328',
'HG00329',
'HG00330',
'HG00331',
'HG00332',
'HG00334',
'HG00335',
'HG00336',
'HG00337',
'HG00338',
'HG00339',
'HG00341',
'HG00342',
'HG00343',
'HG00344',
'HG00345',
'HG00346',
'HG00349',
'HG00350',
'HG00351',
'HG00353',
'HG00355',
'HG00356',
'HG00357',
'HG00358',
'HG00360',
'HG00361',
'HG00362',
'HG00364',
'HG00365',
'HG00366',
'HG00367',
'HG00368',
'HG00369',
'HG00371',
'HG00372',
'HG00373',
'HG00375',
'HG00376',
'HG00378',
'HG00379',
'HG00380',
'HG00381',
'HG00382',
'HG00383',
'HG00384',
'HG00403',
'HG00404',
'HG00406',

'HG00407',
'HG00409',
'HG00410',
'HG00419',
'HG00421',
'HG00422',
'HG00428',
'HG00436',
'HG00437',
'HG00442',
'HG00443',
'HG00445',
'HG00446',
'HG00448',
'HG00449',
'HG00451',
'HG00452',
'HG00457',
'HG00458',
'HG00463',
'HG00464',
'HG00472',
'HG00473',
'HG00475',
'HG00476',
'HG00478',
'HG00479',
'HG00500',
'HG00513',
'HG00524',
'HG00525',
'HG00530',
'HG00531',
'HG00533',
'HG00534',
'HG00536',
'HG00537',
'HG00542',
'HG00543',
'HG00551',
'HG00553',
'HG00554',
'HG00556',
'HG00557',
'HG00559',
'HG00560',
'HG00565',
'HG00566',
'HG00580',
'HG00581',
'HG00583',
'HG00584',
'HG00589',
'HG00590',
'HG00592',
'HG00593',
'HG00595',
'HG00596',
'HG00598',
'HG00599',
'HG00607',
'HG00608',
'HG00610',
'HG00611',
'HG00613',
'HG00614',

'HG00619',
'HG00620',
'HG00622',
'HG00623',
'HG00625',
'HG00626',
'HG00628',
'HG00629',
'HG00631',
'HG00632',
'HG00634',
'HG00637',
'HG00638',
'HG00640',
'HG00641',
'HG00650',
'HG00651',
'HG00653',
'HG00654',
'HG00656',
'HG00657',
'HG00662',
'HG00663',
'HG00671',
'HG00672',
'HG00674',
'HG00675',
'HG00683',
'HG00684',
'HG00689',
'HG00690',
'HG00692',
'HG00693',
'HG00698',
'HG00699',
'HG00701',
'HG00704',
'HG00705',
'HG00707',
'HG00708',
'HG00717',
'HG00728',
'HG00729',
'HG00731',
'HG00732',
'HG00734',
'HG00736',
'HG00737',
'HG00739',
'HG00740',
'HG00742',
'HG00743',
'HG00759',
'HG00766',
'HG00844',
'HG00851',
'HG00864',
'HG00867',
'HG00879',
'HG00881',
'HG00956',
'HG00978',
'HG00982',
'HG01028',
'HG01029',
'HG01031',

'HG01046',
'HG01047',
'HG01048',
'HG01049',
'HG01051',
'HG01052',
'HG01054',
'HG01055',
'HG01058',
'HG01060',
'HG01061',
'HG01063',
'HG01064',
'HG01066',
'HG01067',
'HG01069',
'HG01070',
'HG01072',
'HG01073',
'HG01075',
'HG01077',
'HG01079',
'HG01080',
'HG01082',
'HG01083',
'HG01085',
'HG01086',
'HG01088',
'HG01089',
'HG01092',
'HG01094',
'HG01095',
'HG01097',
'HG01098',
'HG01101',
'HG01102',
'HG01104',
'HG01105',
'HG01107',
'HG01108',
'HG01110',
'HG01111',
'HG01112',
'HG01113',
'HG01119',
'HG01121',
'HG01122',
'HG01124',
'HG01125',
'HG01130',
'HG01131',
'HG01133',
'HG01134',
'HG01136',
'HG01137',
'HG01139',
'HG01140',
'HG01142',
'HG01148',
'HG01149',
'HG01161',
'HG01162',
'HG01164',
'HG01167',
'HG01168',
'HG01170',

'HG01171',
'HG01173',
'HG01174',
'HG01176',
'HG01177',
'HG01182',
'HG01183',
'HG01187',
'HG01188',
'HG01190',
'HG01191',
'HG01197',
'HG01198',
'HG01200',
'HG01204',
'HG01205',
'HG01241',
'HG01242',
'HG01247',
'HG01248',
'HG01250',
'HG01251',
'HG01253',
'HG01254',
'HG01256',
'HG01257',
'HG01259',
'HG01260',
'HG01269',
'HG01271',
'HG01272',
'HG01275',
'HG01277',
'HG01280',
'HG01281',
'HG01284',
'HG01286',
'HG01302',
'HG01303',
'HG01305',
'HG01308',
'HG01311',
'HG01312',
'HG01323',
'HG01325',
'HG01326',
'HG01334',
'HG01341',
'HG01342',
'HG01344',
'HG01345',
'HG01348',
'HG01350',
'HG01351',
'HG01353',
'HG01354',
'HG01356',
'HG01357',
'HG01359',
'HG01360',
'HG01362',
'HG01363',
'HG01365',
'HG01366',
'HG01369',
'HG01372',

'HG01374',
'HG01375',
'HG01377',
'HG01378',
'HG01383',
'HG01384',
'HG01389',
'HG01390',
'HG01392',
'HG01393',
'HG01395',
'HG01396',
'HG01398',
'HG01402',
'HG01403',
'HG01405',
'HG01412',
'HG01413',
'HG01414',
'HG01431',
'HG01432',
'HG01435',
'HG01437',
'HG01438',
'HG01440',
'HG01441',
'HG01443',
'HG01444',
'HG01447',
'HG01455',
'HG01456',
'HG01459',
'HG01461',
'HG01462',
'HG01464',
'HG01465',
'HG01468',
'HG01474',
'HG01479',
'HG01485',
'HG01486',
'HG01488',
'HG01489',
'HG01491',
'HG01492',
'HG01494',
'HG01495',
'HG01497',
'HG01498',
'HG01500',
'HG01501',
'HG01503',
'HG01504',
'HG01506',
'HG01507',
'HG01509',
'HG01510',
'HG01512',
'HG01513',
'HG01515',
'HG01516',
'HG01518',
'HG01519',
'HG01521',
'HG01522',
'HG01524',

'HG01525',
'HG01527',
'HG01528',
'HG01530',
'HG01531',
'HG01536',
'HG01537',
'HG01550',
'HG01551',
'HG01556',
'HG01565',
'HG01566',
'HG01571',
'HG01572',
'HG01577',
'HG01578',
'HG01583',
'HG01586',
'HG01589',
'HG01593',
'HG01595',
'HG01596',
'HG01597',
'HG01598',
'HG01599',
'HG01600',
'HG01602',
'HG01603',
'HG01605',
'HG01606',
'HG01607',
'HG01608',
'HG01610',
'HG01612',
'HG01613',
'HG01615',
'HG01617',
'HG01618',
'HG01619',
'HG01620',
'HG01623',
'HG01624',
'HG01625',
'HG01626',
'HG01628',
'HG01630',
'HG01631',
'HG01632',
'HG01668',
'HG01669',
'HG01670',
'HG01672',
'HG01673',
'HG01675',
'HG01676',
'HG01678',
'HG01679',
'HG01680',
'HG01682',
'HG01684',
'HG01685',
'HG01686',
'HG01694',
'HG01695',
'HG01697',
'HG01699',

'HG01700',
'HG01702',
'HG01704',
'HG01705',
'HG01707',
'HG01708',
'HG01709',
'HG01710',
'HG01746',
'HG01747',
'HG01756',
'HG01757',
'HG01761',
'HG01762',
'HG01765',
'HG01766',
'HG01767',
'HG01768',
'HG01770',
'HG01771',
'HG01773',
'HG01775',
'HG01776',
'HG01777',
'HG01779',
'HG01781',
'HG01783',
'HG01784',
'HG01785',
'HG01786',
'HG01789',
'HG01790',
'HG01791',
'HG01794',
'HG01795',
'HG01796',
'HG01797',
'HG01798',
'HG01799',
'HG01800',
'HG01801',
'HG01802',
'HG01804',
'HG01805',
'HG01806',
'HG01807',
'HG01808',
'HG01809',
'HG01810',
'HG01811',
'HG01812',
'HG01813',
'HG01815',
'HG01816',
'HG01817',
'HG01840',
'HG01841',
'HG01842',
'HG01843',
'HG01844',
'HG01845',
'HG01846',
'HG01847',
'HG01848',
'HG01849',
'HG01850',

'HG01851',
'HG01852',
'HG01853',
'HG01855',
'HG01857',
'HG01858',
'HG01859',
'HG01860',
'HG01861',
'HG01862',
'HG01863',
'HG01864',
'HG01865',
'HG01866',
'HG01867',
'HG01868',
'HG01869',
'HG01870',
'HG01871',
'HG01872',
'HG01873',
'HG01874',
'HG01878',
'HG01879',
'HG01880',
'HG01882',
'HG01883',
'HG01885',
'HG01886',
'HG01889',
'HG01890',
'HG01892',
'HG01893',
'HG01894',
'HG01896',
'HG01912',
'HG01914',
'HG01915',
'HG01917',
'HG01918',
'HG01920',
'HG01921',
'HG01923',
'HG01924',
'HG01926',
'HG01927',
'HG01932',
'HG01933',
'HG01935',
'HG01936',
'HG01938',
'HG01939',
'HG01941',
'HG01942',
'HG01944',
'HG01945',
'HG01947',
'HG01948',
'HG01950',
'HG01951',
'HG01953',
'HG01954',
'HG01956',
'HG01958',
'HG01961',
'HG01965',

'HG01967',
'HG01968',
'HG01970',
'HG01971',
'HG01973',
'HG01974',
'HG01976',
'HG01977',
'HG01979',
'HG01980',
'HG01982',
'HG01985',
'HG01986',
'HG01988',
'HG01989',
'HG01990',
'HG01991',
'HG01992',
'HG01997',
'HG02002',
'HG02003',
'HG02006',
'HG02008',
'HG02009',
'HG02010',
'HG02012',
'HG02013',
'HG02014',
'HG02016',
'HG02017',
'HG02019',
'HG02020',
'HG02023',
'HG02025',
'HG02026',
'HG02028',
'HG02029',
'HG02031',
'HG02032',
'HG02035',
'HG02040',
'HG02047',
'HG02048',
'HG02049',
'HG02050',
'HG02051',
'HG02052',
'HG02053',
'HG02054',
'HG02057',
'HG02058',
'HG02060',
'HG02061',
'HG02064',
'HG02067',
'HG02069',
'HG02070',
'HG02072',
'HG02073',
'HG02075',
'HG02076',
'HG02078',
'HG02079',
'HG02081',
'HG02082',
'HG02084',

'HG02085',
'HG02086',
'HG02087',
'HG02088',
'HG02089',
'HG02090',
'HG02095',
'HG02102',
'HG02104',
'HG02105',
'HG02107',
'HG02108',
'HG02111',
'HG02113',
'HG02116',
'HG02121',
'HG02122',
'HG02127',
'HG02128',
'HG02130',
'HG02131',
'HG02133',
'HG02134',
'HG02136',
'HG02137',
'HG02138',
'HG02139',
'HG02140',
'HG02141',
'HG02142',
'HG02143',
'HG02144',
'HG02146',
'HG02147',
'HG02150',
'HG02151',
'HG02152',
'HG02153',
'HG02154',
'HG02155',
'HG02156',
'HG02164',
'HG02165',
'HG02166',
'HG02178',
'HG02179',
'HG02180',
'HG02181',
'HG02182',
'HG02184',
'HG02185',
'HG02186',
'HG02187',
'HG02188',
'HG02190',
'HG02215',
'HG02219',
'HG02220',
'HG02221',
'HG02223',
'HG02224',
'HG02230',
'HG02231',
'HG02232',
'HG02233',
'HG02235',

'HG02236',
'HG02238',
'HG02239',
'HG02250',
'HG02252',
'HG02253',
'HG02255',
'HG02256',
'HG02259',
'HG02260',
'HG02262',
'HG02265',
'HG02266',
'HG02271',
'HG02272',
'HG02274',
'HG02275',
'HG02277',
'HG02278',
'HG02281',
'HG02282',
'HG02283',
'HG02284',
'HG02285',
'HG02286',
'HG02291',
'HG02292',
'HG02298',
'HG02299',
'HG02301',
'HG02304',
'HG02307',
'HG02308',
'HG02309',
'HG02312',
'HG02314',
'HG02315',
'HG02317',
'HG02318',
'HG02322',
'HG02323',
'HG02325',
'HG02330',
'HG02332',
'HG02334',
'HG02337',
'HG02339',
'HG02343',
'HG02345',
'HG02348',
'HG02351',
'HG02353',
'HG02355',
'HG02356',
'HG02360',
'HG02364',
'HG02367',
'HG02371',
'HG02373',
'HG02374',
'HG02375',
'HG02379',
'HG02380',
'HG02382',
'HG02383',
'HG02384',

'HG02385',
'HG02386',
'HG02389',
'HG02390',
'HG02391',
'HG02392',
'HG02394',
'HG02395',
'HG02396',
'HG02397',
'HG02398',
'HG02399',
'HG02401',
'HG02402',
'HG02406',
'HG02407',
'HG02408',
'HG02409',
'HG02410',
'HG02419',
'HG02420',
'HG02425',
'HG02427',
'HG02429',
'HG02433',
'HG02439',
'HG02442',
'HG02445',
'HG02449',
'HG02450',
'HG02455',
'HG02461',
'HG02462',
'HG02464',
'HG02465',
'HG02470',
'HG02471',
'HG02476',
'HG02477',
'HG02479',
'HG02481',
'HG02484',
'HG02485',
'HG02489',
'HG02490',
'HG02491',
'HG02493',
'HG02494',
'HG02496',
'HG02497',
'HG02501',
'HG02502',
'HG02505',
'HG02508',
'HG02511',
'HG02512',
'HG02513',
'HG02521',
'HG02522',
'HG02536',
'HG02537',
'HG02541',
'HG02545',
'HG02546',
'HG02549',
'HG02554',

```
 'HG02555',
 'HG02557',
 'HG02558',
 'HG02561',
 'HG02562',
 'HG02568',
 'HG02570',
 'HG02571',
 'HG02573',
 'HG02574',
 'HG02577',
 ...]
```

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

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

```
vcfSourceFile = pd.read_csv(
    DATASET_PAH + "ALL.wgs.mergedSV.v8.20130502.svs.genotypes.vcf",
    sep="\t",
    comment="#",
    names=columns,
    header=None,
)
```

```
/tmp/ipykernel_20700/2618528307.py:1: DtypeWarning: Columns (0) have mixed types. Specify dtype option
on import or set low_memory=False.
  vcfSourceFile = pd.read_csv(
```

```
vcfSourceFile
```

	CHROM	POS	ID	
0	1	645710	ALU_umary_ALU_2	
1	1	668630	DUP_delly_DUP20532	
2	1	713044	DUP_gs_CNV_1_713044_755966	
3	1	738570	UW_VH_21763	
4	1	766600	UW_VH_5595	
...	...	...	...	
68813	X	155064470	DUP_gs.X_CNV_X_155064470_155081667	
68814	X	155090084	UW_VH_7995	
68815	X	155120139	L1_umary_LINE1_3151	
68816	X	155122541	DEL_pindel_54975	GAGTAACTGGGATGACAGGCGTGTGCCACCACGCCTGG
68817	X	155146395	BI_GS_DEL1_B3_P3053_10	

68818 rows × 2513 columns

```
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"] == "20"].copy()
```

```
vcfSourceFile
```

	CHROM	POS	ID	
63261	20	118471	DUP_delly_DUP55813	
63262	20	118498	DUP_gs_CNV_20_118498_242177	
63263	20	249590	BI_GS_DEL1_B5_P2733_211	
63264	20	283851	SVA_umary_SVA_770	
63265	20	294022	BI_GS_DEL1_B4_P2733_46	
...	...	...	...	
64825	20	62891341	DUP_gs_CNV_20_62891341_62900705	
64826	20	62891722	DEL_pindel_51199	GAGACCGCAGCCTTCCCAGTAGTTATCTGACATTGACACATCA
64827	20	62904563	DUP_gs_CNV_20_62904563_62915351	
64828	20	62949512	DUP_gs_CNV_20_62949512_62957862	
64829	20	62949699	UW_VH_1926	

1569 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"] == "20"].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)
```

```
# Data preprocessing: ensuring proper types, mapping

vcfSourceFile["QUAL"] = vcfSourceFile["QUAL"].apply(lambda x: int(x) if x != "." else 0)

for column in commonColumns:
    vcfSourceFile[column] = vcfSourceFile[column].apply(
        # lambda x: sum([int(n) for n in x.split("|")]) if x != "." else 0
        lambda x: max([int(n) for n in x.split("|")]) if x != "." else 0
    )
for column in commonColumns:
```

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

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


def processRow(row):
    for column in commonColumns:
        if row[column] != 0:
            row[column] = row["ALT"][row[column] - 1]
    return row


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

# 1. DONE: For each row count the number of occurences of each ALT, save it to a dictionary and turn to
a dataframe, and merge based on id with vcf data
# 2. Find a way to merge rna and vcf dataframes, perhaps on closest position
```

```
def countUniqueALTs(row, d):
    uniqueALTs = row[commonColumns].unique()
    rowId = row.name
    d[rowId] = {}
    for alt in uniqueALTs:
        if alt == 0:
            continue
        d[rowId][alt] = 0
        for column in commonColumns:
            if row[column] == alt:
                d[rowId][alt] = d[rowId][alt] + 1
    for alt in d[rowId]:
        row[alt] = d[rowId][alt]
    return row


d = {}
vcfSourceFileCountedAlts = vcfSourceFile.apply(lambda x: countUniqueALTs(x, d), axis=1)
```

```
vcfSourceFileCountedAlts["POS"]
```

```
63261      118471
63262      118498
63263      249590
63264      283851
63265      294022
           ...
64825    62891341
64826    62891722
64827    62904563
64828    62949512
64829    62949699
Name: POS, Length: 1569, dtype: int64
```

```
rnaSeqData.columns
```

```
Index(['TargetID', 'Gene_Symbol', 'Chr', 'Coord', 'HG00096', 'HG00097',
       'HG00099', 'HG00099', 'HG00100', 'HG00101',
       ...
       'NA20810', 'NA20811', 'NA20812', 'NA20813', 'NA20814', 'NA20815',
       'NA20816', 'NA20819', 'NA20826', 'NA20828'],
      dtype='object', length=664)
```

```
rnaSeqData["Coord"]
```

```
17        44144264
66        20348765
90        22946700
120       42418584
183       18040137
            ...
53770     21143531
53900     54967393
53901     30795594
53909     54987168
53926     38660227
Name: Coord, Length: 1276, dtype: int64
```

```python
def mergeRnaOnClosest(row, rna):
    closestRna = {"closest": -1, "distance": np.inf}

    def findClosestRna(rnaRow, closestRna):
        distance = np.abs(row["POS"] - rnaRow["Coord"])
        if distance < closestRna["distance"]:
            closestRna["closest"] = rnaRow

    rna.apply(lambda x: findClosestRna(x, closestRna), axis=1)
    # Merge on closestRna
    return pd.concat([row, closestRna["closest"]], axis=0)


rnaSeqData["mean"] = rnaSeqData[commonColumns].mean(axis=1)
vcfRnaMerged = vcfSourceFileCountedAlts.apply(
    lambda x: mergeRnaOnClosest(x, rnaSeqData), axis=1
).reset_index()
```

```python
vcfRnaMerged
```

	index	<CN0>	<CN2>	<CN3>	<INS:ME:ALU>	<INS:ME:LINE1>	<INS:ME:SVA>	<INS:MT>	<INV>	A
0	63261	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	63262	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	63263	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	63264	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	63265	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
1564	64825	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1565	64826	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1566	64827	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1567	64828	114.0	27.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN
1568	64829	73.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1569 rows × 3191 columns

```python
vcfSourceFile["HG00108"].unique()
```

```
array([0, 'T', '<CN0>', 'C', '<INS:ME:SVA>', 'A', '<INS:ME:LINE1>',
       '<INS:ME:ALU>', 'G', '<INV>'], dtype=object)
```

```
vcfFilteredQuality = vcfSourceFile[vcfSourceFile["QUAL"] > 90]
```

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

```
vcfSourceFile[vcfBaseColumns + list(commonColumns)][vcfSourceFile["CHROM"] == "20"]
```

	CHROM	POS	ID	
63261	20	118471	DUP_delly_DUP55813	
63262	20	118498	DUP_gs_CNV_20_118498_242177	
63263	20	249590	BI_GS_DEL1_B5_P2733_211	
63264	20	283851	SVA_umary_SVA_770	
63265	20	294022	BI_GS_DEL1_B4_P2733_46	
...	...	...	...	
64825	20	62891341	DUP_gs_CNV_20_62891341_62900705	
64826	20	62891722	DEL_pindel_51199	GAGACCGCAGCCTTCCCAGTAGTTATCTGACATTGACACATCA
64827	20	62904563	DUP_gs_CNV_20_62904563_62915351	
64828	20	62949512	DUP_gs_CNV_20_62949512_62957862	
64829	20	62949699	UW_VH_1926	

1569 rows × 454 columns

```
rnaSeqData[rnaBaseColumns + list(commonColumns)][rnaSeqData["Chr"] == "20"]
```

	TargetID	Gene_Symbol	Chr	Coord	HG00096	HG00097	HG00099	HG00099	HG00
17	ENSG00000101446.7	ENSG00000101446.7	20	44144264	0.00000	0.00000	0.00000	0.00000	0.0(
66	ENSG00000173404.3	ENSG00000173404.3	20	20348765	0.01355	0.00000	0.01756	0.00000	0.0(
90	ENSG00000228935.1	ENSG00000228935.1	20	22946700	0.00000	0.00000	0.00000	0.00000	0.0(
120	ENSG00000225865.1	ENSG00000225865.1	20	42418584	0.01580	0.00995	0.00000	0.00000	0.0(
183	ENSG00000229262.1	ENSG00000229262.1	20	18040137	0.00000	0.00000	0.00000	0.00000	0.0(
...	...	...	...	...	...	...	...	...	
53770	ENSG00000228604.1	ENSG00000228604.1	20	21143531	0.00000	0.00000	0.00000	0.00000	0.0(
53900	ENSG00000087586.13	ENSG00000087586.13	20	54967393	16.44485	16.66117	28.56229	15.08591	26.0
53901	ENSG00000126003.6	ENSG00000126003.6	20	30795594	6.99132	9.12025	10.44336	6.08358	8.1
53909	ENSG00000087589.12	ENSG00000087589.12	20	54987168	0.04079	0.10652	0.07583	0.02172	0.1(
53926	ENSG00000233415.1	ENSG00000233415.1	20	38660227	0.00000	0.00000	0.00000	0.04445	0.0(

1276 rows × 642 columns

Perform linear regression

```
altColumns = []
for column in list(vcfRnaMerged.columns):
    if "<" in column:
        altColumns.append(column)
```

```
altColumns
```

```
['<CN0>',
 '<CN2>',
 '<CN3>',
 '<INS:ME:ALU>',
 '<INS:ME:LINE1>',
 '<INS:ME:SVA>',
 '<INS:MT>',
 '<INV>']
```

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

```
array([0.0005613])
```

```
# regression on '<CN0>', '<CN2>', '<CN3>',
# on '<INS:ME:ALU>', '<INS:ME:LINE1>', '<INS:ME:SVA>', '<INS:MT>'
# on '<INV>'


mod = statsmodels.api.OLS(
    vcfRnaMerged["mean"], vcfRnaMerged[["<CN0>", "<CN2>", "<CN3>"]].fillna(value=0)
)
fii = mod.fit()
p_values1 = fii.summary2().tables[1]["P>|t|"]


mod = statsmodels.api.OLS(
    vcfRnaMerged["mean"],
    vcfRnaMerged[["<INS:ME:ALU>", "<INS:ME:LINE1>", "<INS:ME:SVA>", "<INS:MT>"]].fillna(
```

```
        value=0
    ),
)
fii = mod.fit()
p_values2 = fii.summary2().tables[1]["P>|t|"]


mod = statsmodels.api.OLS(vcfRnaMerged["mean"], vcfRnaMerged["<INV>"].fillna(value=0))
fii = mod.fit()
p_values3 = fii.summary2().tables[1]["P>|t|"]

mod = statsmodels.api.OLS(
    vcfRnaMerged["mean"], vcfRnaMerged[["A", "C", "T", "G"]].fillna(value=0)
)
fii = mod.fit()
p_values4 = fii.summary2().tables[1]["P>|t|"]
```

```
p_values1
```

```
<CN0>    1.897710e-09
<CN2>    6.578281e-03
<CN3>    1.606035e-01
Name: P>|t|, dtype: float64
```

```
p_values2
```

```
<INS:ME:ALU>      4.627195e-09
<INS:ME:LINE1>    1.795547e-02
<INS:ME:SVA>      3.305906e-02
<INS:MT>          3.110874e-01
Name: P>|t|, dtype: float64
```

```
p_values3
```

```
<INV>    0.311753
Name: P>|t|, dtype: float64
```

```
p_values4
```

```
A    0.002040
C    0.003130
T    0.000010
G    0.009612
Name: P>|t|, dtype: float64
```

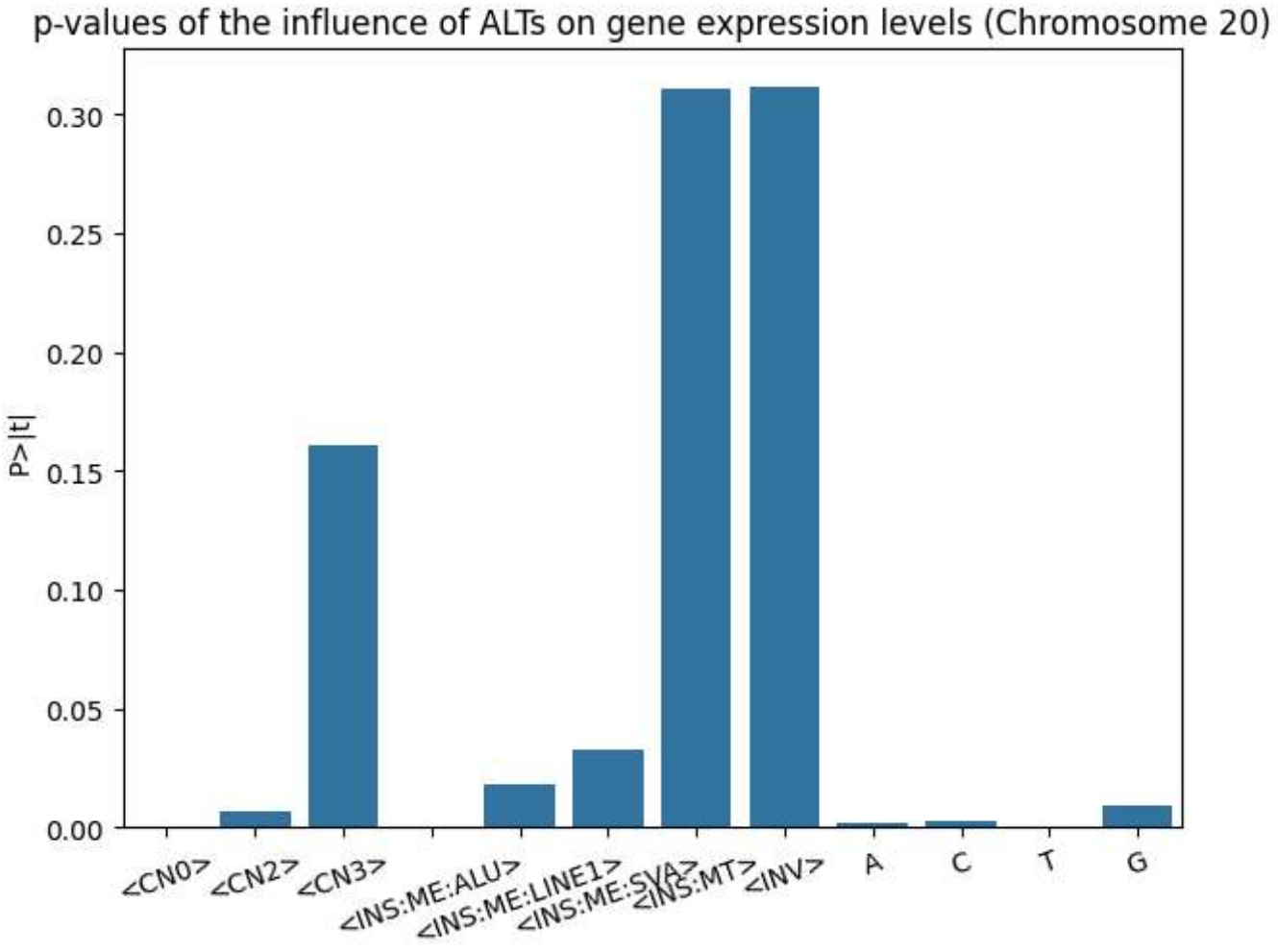
Represent the results

```
# p-values for the effect of different ALT values on gene expression levels

print(p_values1, "\n", p_values2, "\n", p_values3, "\n", p_values4)
```

```
<CN0>    1.897710e-09
<CN2>    6.578281e-03
<CN3>    1.606035e-01
Name: P>|t|, dtype: float64
 <INS:ME:ALU>      4.627195e-09
<INS:ME:LINE1>    1.795547e-02
<INS:ME:SVA>      3.305906e-02
<INS:MT>          3.110874e-01
Name: P>|t|, dtype: float64
 <INV>    0.311753
Name: P>|t|, dtype: float64
 A    0.002040
C    0.003130
T    0.000010
G    0.009612
Name: P>|t|, dtype: float64
```

```
plt.title("p-values of the influence of ALTs on gene expression levels (Chromosome 20)")
sns.barplot(pd.concat([p_values1, p_values2, p_values3, p_values4], axis=0))
plt.tight_layout()
plt.xticks(rotation=20)
plt.show()
```




```
vcfRnaMerged.to_csv("mergedData.csv")
```