

```
In [ ]: import sys
        !{sys.executable} -m pip install --user scikit-allel
```

```
In [1]: import numpy as np
import scipy
import pandas
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('white')
sns.set_style('ticks')
sns.set_context('notebook')
import h5py
import allel; print('scikit-allel', allel.__version__)
```

scikit-allel 1.3.8

VCF to HDF5

```
In [2]: #allel.vcf_to_hdf5('/users/mcevoysu/scratch/output/Phalepensis/vcf_filter
```

Get data

```
In [2]: callset_var_fn = '/users/mcevoysu/scratch/output/Phalepensis/scikit-allel
callset_var = h5py.File(callset_var_fn, mode='r')
```

```
In [3]: calldata_var = callset_var['calldata']
list(calldata_var)
```

```
Out[3]: ['AD', 'DP', 'GQ', 'GT', 'MIN_DP', 'PGT', 'PID', 'PL', 'PS', 'RGQ', 'S
B']
```

```
In [4]: list(callset_var['variants'])
```

```
Out [4]: ['AC',
          'AF',
          'ALT',
          'AN',
          'BaseQRankSum',
          'CHROM',
          'DP',
          'END',
          'ExcessHet',
          'FILTER_LowQual',
          'FILTER_PASS',
          'FS',
          'ID',
          'InbreedingCoeff',
          'MLEAC',
          'MLEAF',
          'MQ',
          'MQRankSum',
          'POS',
          'QD',
          'QUAL',
          'RAW_MQandDP',
          'REF',
          'ReadPosRankSum',
          'SOR',
          'altlen',
          'is_snp',
          'numalt']
```

Make datasets

```
In [5]: variants = allel.VariantChunkedTable(callset_var['variants'])
variants
```

```
Out [5]: <VariantChunkedTable shape=(331610,) dtype=[('AC', '<i4', (3,)), ('AF', '<f4', (3,)),
('ALT', 'O', (3,)), ('AN', '<i4'), ('BaseQRankSum', '<f4'), ('CHROM', 'O'), ('DP', '<i4'),
('END', '<i4'), ('ExcessHet', '<f4'), ('FILTER_LowQual', '?'), ('FILTER_PASS', '?'), ('FS',
'<f4'), ('ID', 'O'), ('InbreedingCoeff', '<f4'), ('MLEAC', '<i4', (3,)), ('MLEAF', '<f4', (3,)),
('MQ', '<f4'), ('MQRankSum', '<f4'), ('POS', '<i4'), ('QD', '<f4'), ('QUAL', '<f4'),
('RAW_MQandDP', '<i4', (2,)), ('REF', 'O'), ('ReadPosRankSum', '<f4'), ('SOR', '<f4'),
('altlen', '<i4', (3,)), ('is_snp', '?'), ('numalt', '<i4')] nbytes=56.6M cbytes=12.4M
cratio=4.6 values=h5py._hl.group.Group>
```

	AC	AF	ALT	AN	BaseQRankSum	CHROM	DP	ENI
0	[2 -1 -1]	[0.002674 nan nan]	[b'*' b'' b'']	742	nan	b'chr1.c3'	5697	-1
1	[1 -1 -1]	[0.001337 nan nan]	[b'*' b'' b'']	742	-0.565	b'chr1.c3'	5461	-1
2	[5 -1 -1]	[0.006684 nan nan]	[b'G' b'' b'']	742	-0.74	b'chr1.c3'	5486	-1
...								
331607	[1 -1 -1]	[0.001337 nan nan]	[b'G' b'' b'']	742	-0.224	b'tig00272805_1_2'	6313	-1
331608	[1 -1 -1]	[0.001337 nan nan]	[b'A' b'' b'']	742	0.0	b'tig00272805_1_2'	6317	-1
331609	[2 -1 -1]	[0.002674 nan nan]	[b'C' b'' b'']	742	-2.1	b'tig00272805_1_2'	6566	-1

```
In [6]: variants_np = variants[:,]
rawsnps = variants_np.query('(is_snp == True)')
rawsnps
```

```
Out [6]: <VariantTable shape=(249352,) dtype=(numpy.record, [('AC', '<i4', (3,)), ('AF', '<f4', (3,)), ('ALT', 'O', (3,)), ('AN', '<i4', (3,)), ('BaseQRankSum', '<f4', (3,)), ('CHROM', 'O', (3,)), ('DP', '<i4', (3,)), ('END', '<i4', (3,)), ('ExcessHet', '<f4', (3,)), ('FILTER_LowQual', '?', (3,)), ('FILTER_PASS', '?', (3,)), ('FS', '<f4', (3,)), ('ID', 'O', (3,)), ('InbreedingCoeff', '<f4', (3,)), ('MLEAC', '<i4', (3,)), ('MLEAF', '<f4', (3,)), ('MQ', '<f4', (3,)), ('MQRankSum', '<f4', (3,)), ('POS', '<i4', (3,)), ('QD', '<f4', (3,)), ('QUAL', '<f4', (3,)), ('RAW_MQandDP', '<i4', (2,)), ('REF', 'O', (3,)), ('ReadPosRankSum', '<f4', (3,)), ('SOR', '<f4', (3,)), ('altlen', '<i4', (3,)), ('is_snp', '?', (3,)), ('numalt', '<i4', (3,))])>
```

	AC	AF	ALT	AN	BaseQRankSum	CHROM	DP	...
0	[5 -1 -1]	[0.006684 nan nan]	[b'G' b'' b'']	742	-0.74	b'chr1.c3'	5486	.
1	[697 -1 -1]	[0.94 nan nan]	[b'T' b'' b'']	742	0.403	b'chr1.c3'	5363	.
2	[622 -1 -1]	[0.84 nan nan]	[b'C' b'' b'']	742	nan	b'chr1.c3'	1690	.
...								
249349	[1 -1 -1]	[0.001337 nan nan]	[b'G' b'' b'']	742	-0.224	b'tig00272805_1_2'	6313	.
249350	[1 -1 -1]	[0.001337 nan nan]	[b'A' b'' b'']	742	0.0	b'tig00272805_1_2'	6317	.
249351	[2 -1 -1]	[0.002674 nan nan]	[b'C' b'' b'']	742	-2.1	b'tig00272805_1_2'	6566	.

```
In [7]: notsnp = variants_np.query('(is_snp != True)')
notsnp
```

```
Out [7]: <VariantTable shape=(82258,) dtype=(numpy.record, [('AC', '<i4', (3,)), ('AF', '<f4', (3,)), ('ALT', 'O', (3,)), ('AN', '<i4'), ('BaseQRankSum', '<f4'), ('CHROM', 'O'), ('DP', '<i4'), ('END', '<i4'), ('ExcessHet', '<f4'), ('FILTER_LowQual', '?'), ('FILTER_PASS', '?'), ('FS', '<f4'), ('ID', 'O'), ('InbreedingCoeff', '<f4'), ('MLEAC', '<i4', (3,)), ('MLEAF', '<f4', (3,)), ('MQ', '<f4'), ('MQRankSum', '<f4'), ('POS', '<i4'), ('QD', '<f4'), ('QUAL', '<f4'), ('RAW_MQandDP', '<i4', (2,)), ('REF', 'O'), ('ReadPosRankSum', '<f4'), ('SOR', '<f4'), ('altlen', '<i4', (3,)), ('is_snp', '?'), ('numalt', '<i4')])>
```

	AC	AF	ALT	AN	BaseQRankSum	CHROM	DP	END
0	[2 -1 -1]	[0.002674 nan nan]	[b'*' b'' b'']	742	nan	b'chr1.c3'	5697	-1
1	[1 -1 -1]	[0.001337 nan nan]	[b'*' b'' b'']	742	-0.565	b'chr1.c3'	5461	-1
2	[1 -1 -1]	[0.001337 nan nan]	[b'*' b'' b'']	742	nan	b'chr1.c3'	1289	-1
...								
82255	[21 -1 -1]	[0.028 nan nan]	[b'*' b'' b'']	740	nan	b'tig10501301_1_2'	600	-1
82256	[11 -1 -1]	[0.015 nan nan]	[b'*' b'' b'']	742	nan	b'tig10501301_1_2'	2254	-1
82257	[2 -1 -1]	[0.002674 nan nan]	[b'*' b'' b'']	742	-0.387	b'tig00277599'	3397	-1

Plot function

```
In [8]: def plot_hist(f, dsubset='', bins=30, ):
    if dsubset == 'var':
        x = variants[f][:]
        l = 'Variant'
    elif dsubset == 'snp':
        x = rawsnps[f][:]
        l = 'Raw SNP'
    elif dsubset == 'notsnp':
        x = notsnp[f][:]
        l = 'Raw Not SNP'
    elif dsubset == 'biallelic':
        x = biallelic_np[f][:]
        l = 'Biallelic SNP'
    elif dsubset == 'varsel':
        x = var_selection[f][:]
        l = 'Filtered Variants'
    elif dsubset == 'snpsel':
        x = snp_selection[f][:]
        l = 'Filtered SNP'
    else:
```

```

        x = bi_selection[f][:]
        l = 'Biallelic SNP'
    fig, ax = plt.subplots(figsize=(10, 5))
    sns.despine(ax=ax, offset=10)
    ax.hist(x, bins=bins)
    ax.set_xlabel(f)
    ax.set_ylabel('No. variants')
    ax.set_title('%s %s distribution' % (l, f))

```

Find Biallelic SNPS

```

In [9]: numalt = rawsnps['numalt']
        np.max(numalt)

```

Out[9]: 3

```

In [10]: count_numalt = np.bincount(numalt)
         count_numalt

```

Out[10]: array([0, 242354, 6824, 174])

```

In [11]: n_multiallelic = np.sum(count_numalt[2:])
         n_multiallelic

```

Out[11]: 6998

```

In [12]: filter_expression = '(numalt == 1)'
         biallelic_np = rawsnps.query(filter_expression)[: ]
         biallelic_np

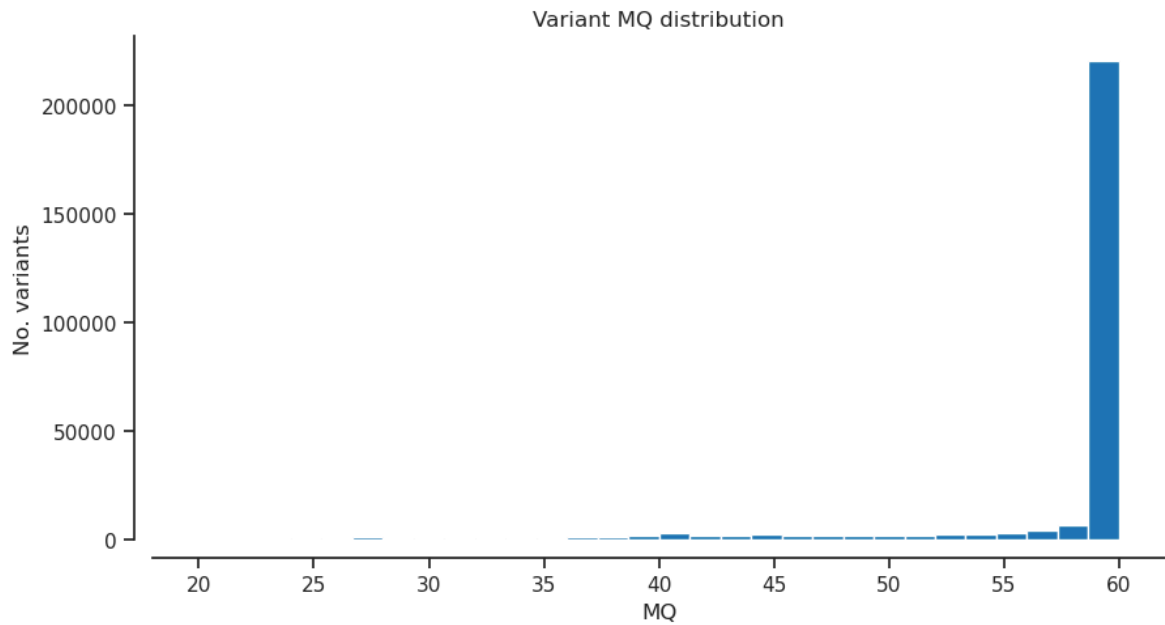
```

```
Out [12]: <VariantTable shape=(242354,) dtype=(numpy.record, [('AC', '<i4', (3,)), ('AF', '<f4', (3,)), ('ALT', 'O', (3,)), ('AN', '<i4', ('BaseQRankSum', '<f4'), ('CHROM', 'O'), ('DP', '<i4'), ('END', '<i4'), ('ExcessHet', '<f4'), ('FILTER_LowQual', '?'), ('FILTER_PASS', '?'), ('FS', '<f4'), ('ID', 'O'), ('InbreedingCoeff', '<f4'), ('MLEAC', '<i4', (3,)), ('MLEAF', '<f4', (3,)), ('MQ', '<f4'), ('MQRankSum', '<f4'), ('POS', '<i4'), ('QD', '<f4'), ('QUAL', '<f4'), ('RAW_MQandDP', '<i4', (2,)), ('REF', 'O'), ('ReadPosRankSum', '<f4'), ('SOR', '<f4'), ('altlen', '<i4', (3,)), ('is_snp', '?'), ('numalt', '<i4')]))>
```

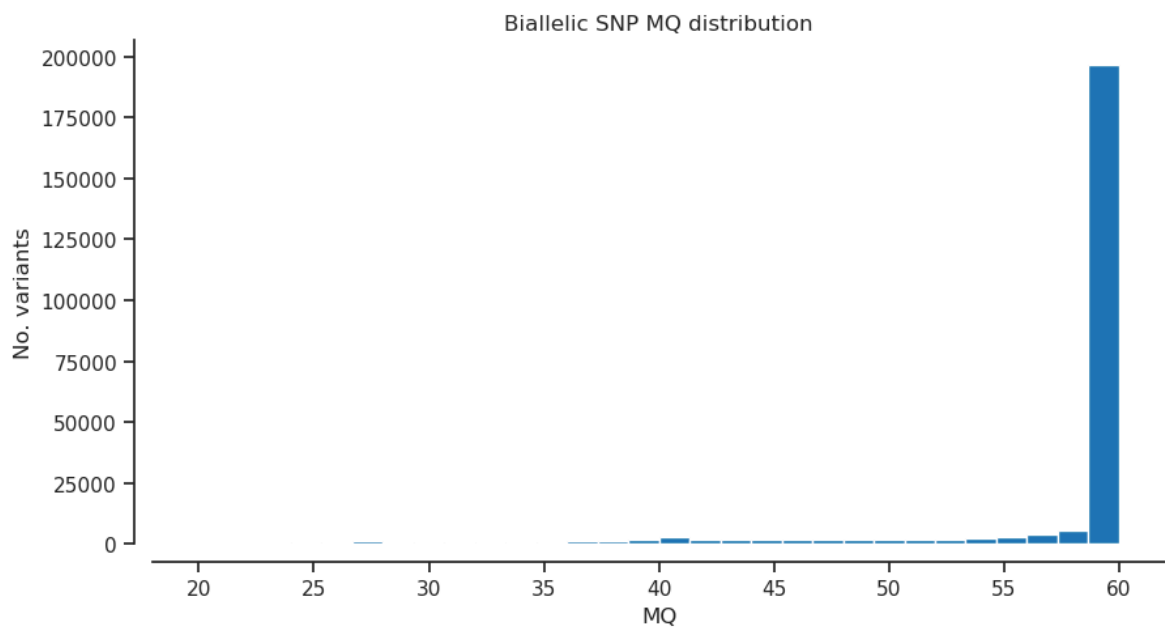
	AC	AF	ALT	AN	BaseQRankSum	CHROM	DP	EQ
0	[5 -1 -1]	[0.006684 nan nan]	[b'G' b'' b'']	742	-0.74	b'chr1.c3'	5486	-
1	[697 -1 -1]	[0.94 nan nan]	[b'T' b'' b'']	742	0.403	b'chr1.c3'	5363	-
2	[622 -1 -1]	[0.84 nan nan]	[b'C' b'' b'']	742	nan	b'chr1.c3'	1690	-
...								
242351	[1 -1 -1]	[0.001337 nan nan]	[b'G' b'' b'']	742	-0.224	b'tig00272805_1_2'	6313	-
242352	[1 -1 -1]	[0.001337 nan nan]	[b'A' b'' b'']	742	0.0	b'tig00272805_1_2'	6317	-
242353	[2 -1 -1]	[0.002674 nan nan]	[b'C' b'' b'']	742	-2.1	b'tig00272805_1_2'	6566	-

MQ - RMS mapping quality

```
In [13]: plot_hist('MQ', 'var') # RMS mapping quality
```

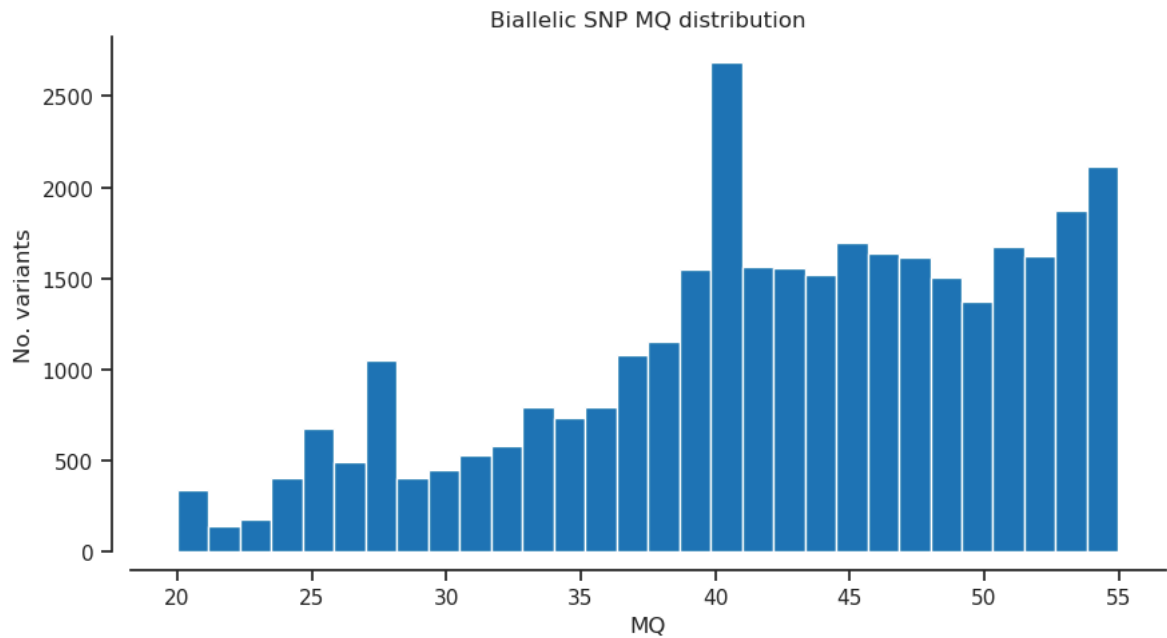


```
In [14]: plot_hist('MQ','biallelic') # RMS mapping quality
```



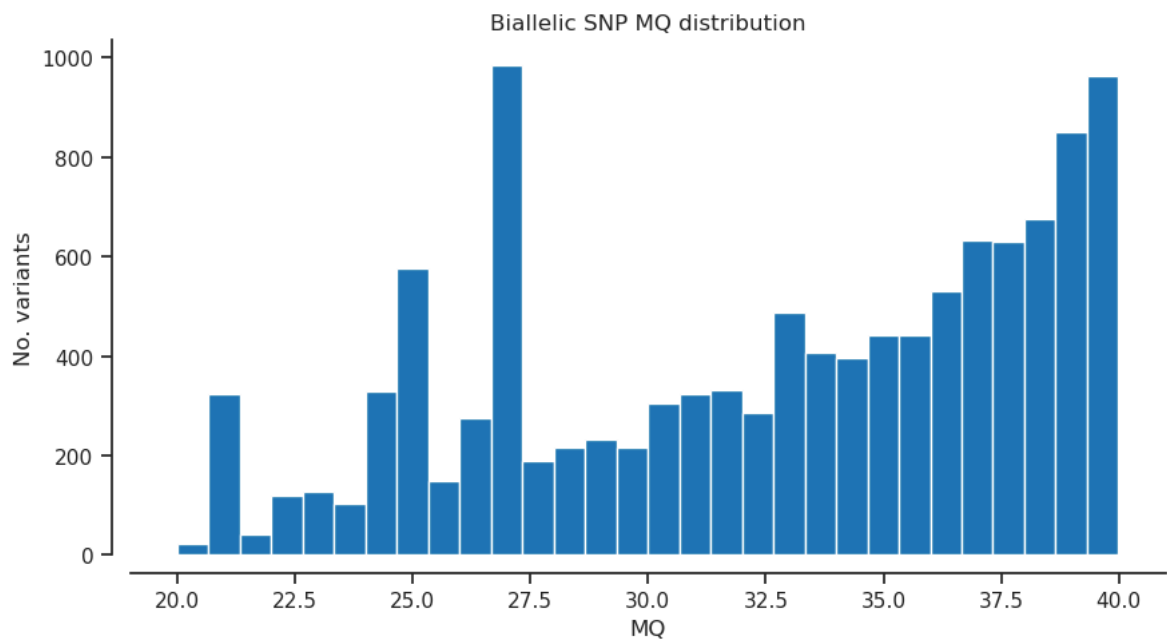
```
In [15]: filter_expression = '(MQ < 55)'
bi_selection = biallelic_np.query(filter_expression)[: ]
#np.count_nonzero(var_selection)
```

```
In [16]: plot_hist('MQ')
```

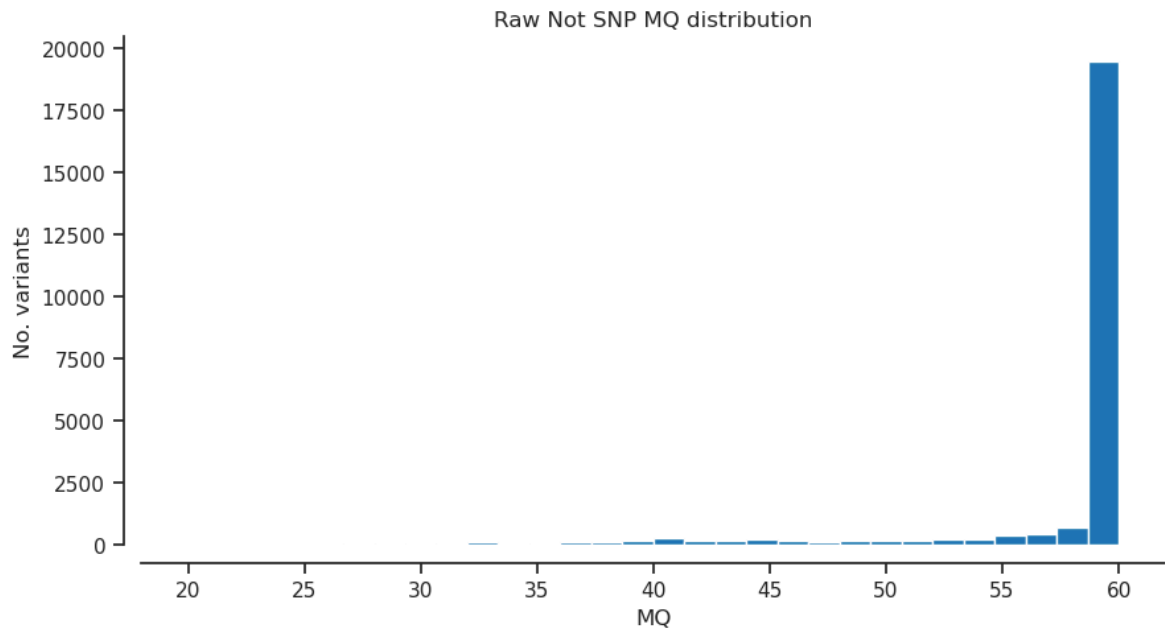



```
In [17]: filter_expression = '(MQ < 40)'  
bi_selection = biallelic_np.query(filter_expression)[:]
```

```
In [18]: plot_hist('MQ')
```

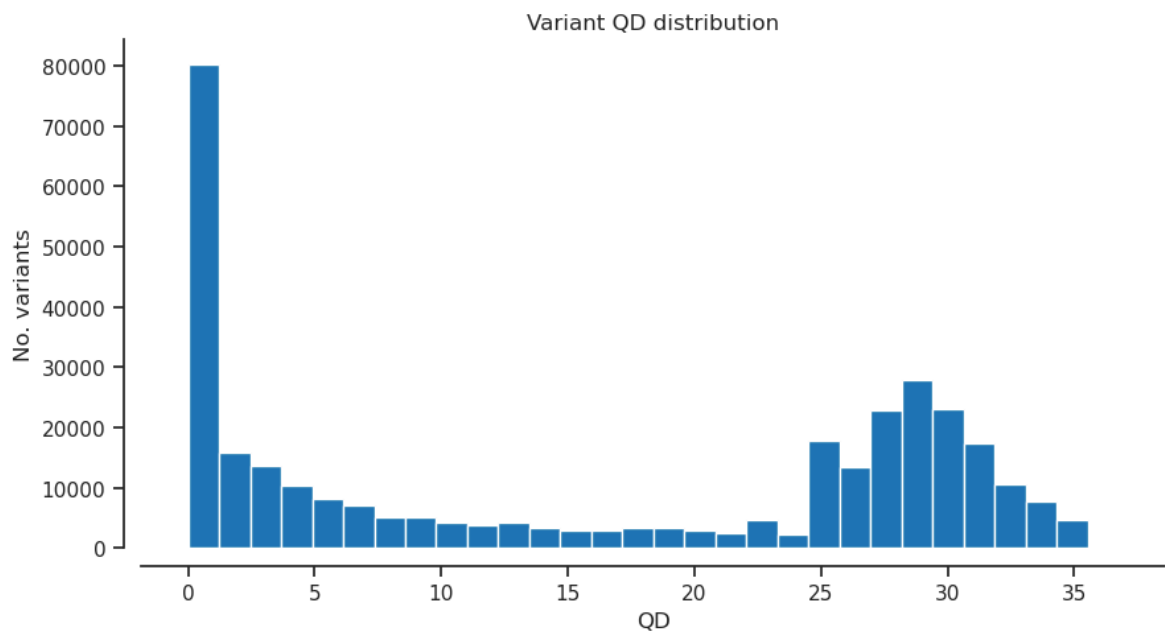


```
In [19]: plot_hist('MQ', 'notsnp')
```

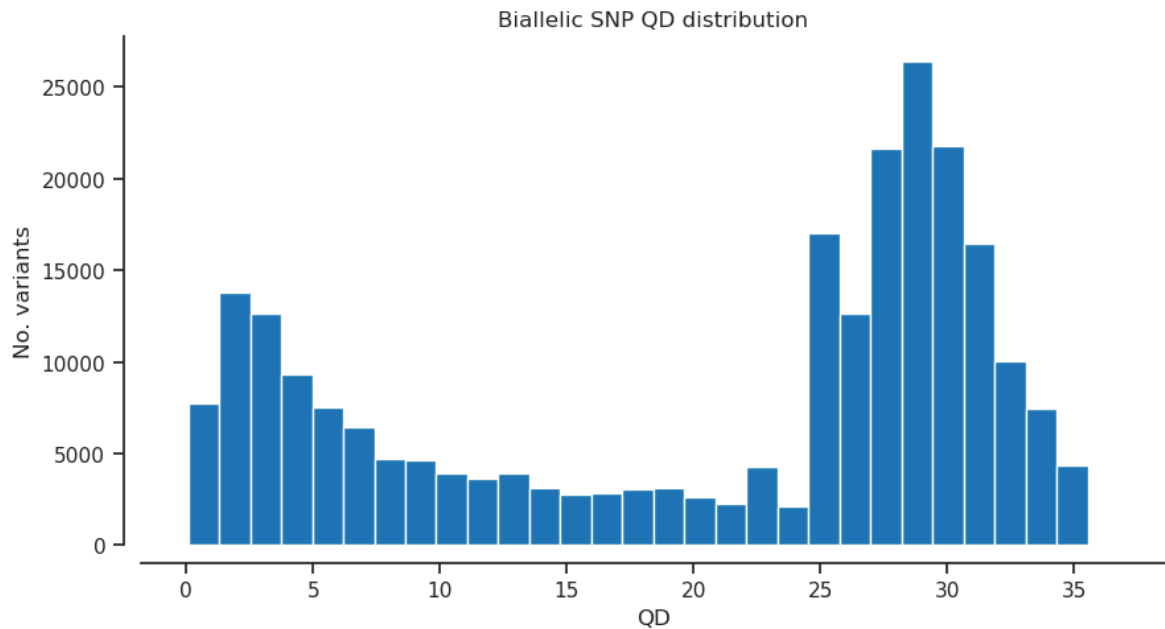


QD - Variant Confidence/Quality by Depth

```
In [20]: plot_hist('QD','var') # Variant Confidence/Quality by Depth
```

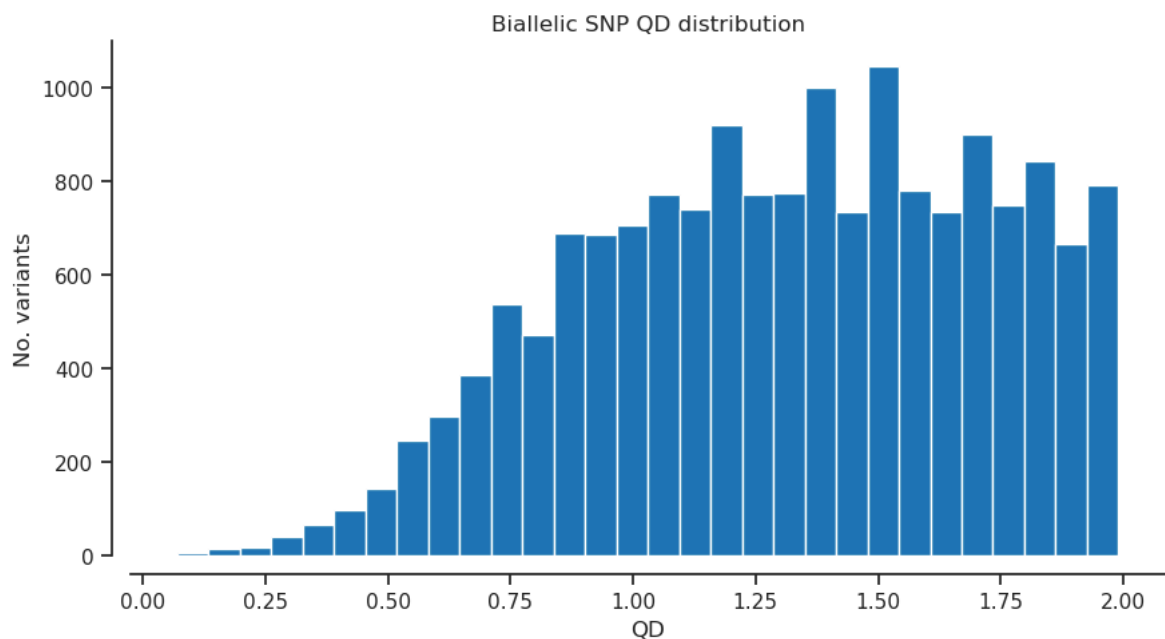


```
In [21]: plot_hist('QD','biallelic') # Variant Confidence/Quality by Depth
```

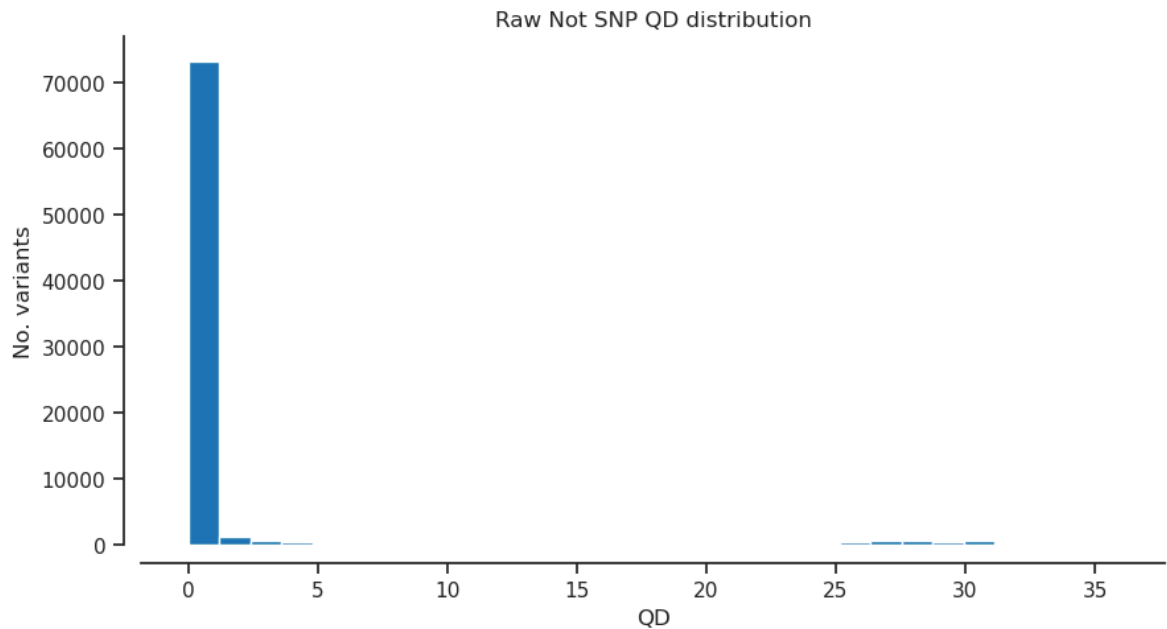


```
In [22]: filter_expression = '(QD < 2)'  
bi_selection = biallelic_np.query(filter_expression)[:]
```

```
In [23]: plot_hist('QD')
```

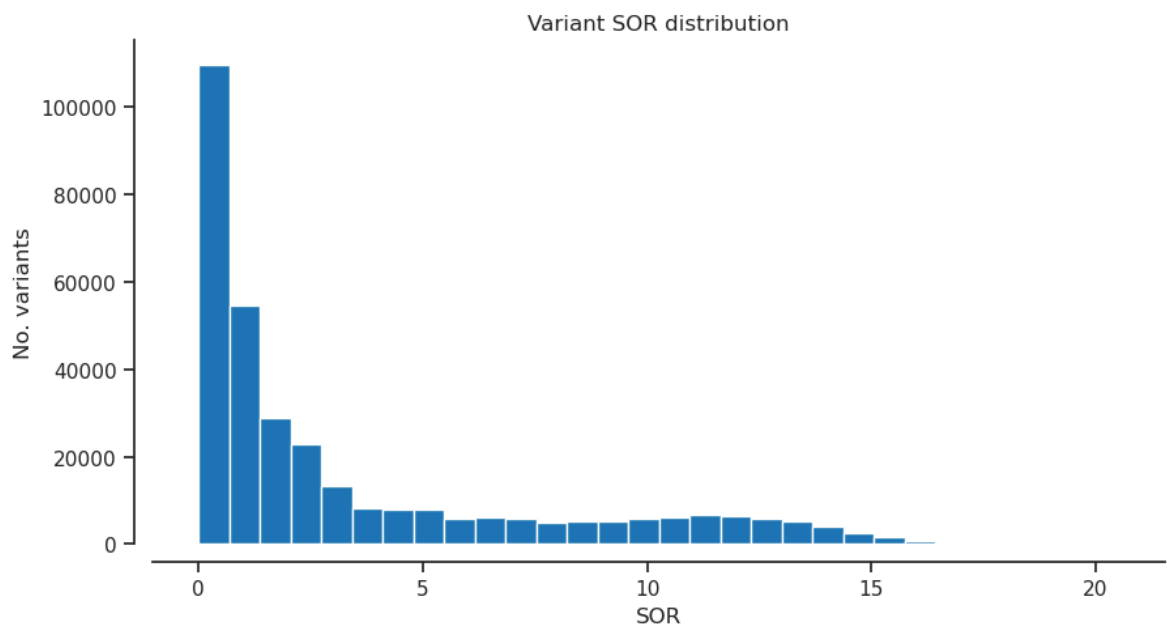


```
In [24]: plot_hist('QD','notsnp') # Variant Confidence/Quality by Depth
```

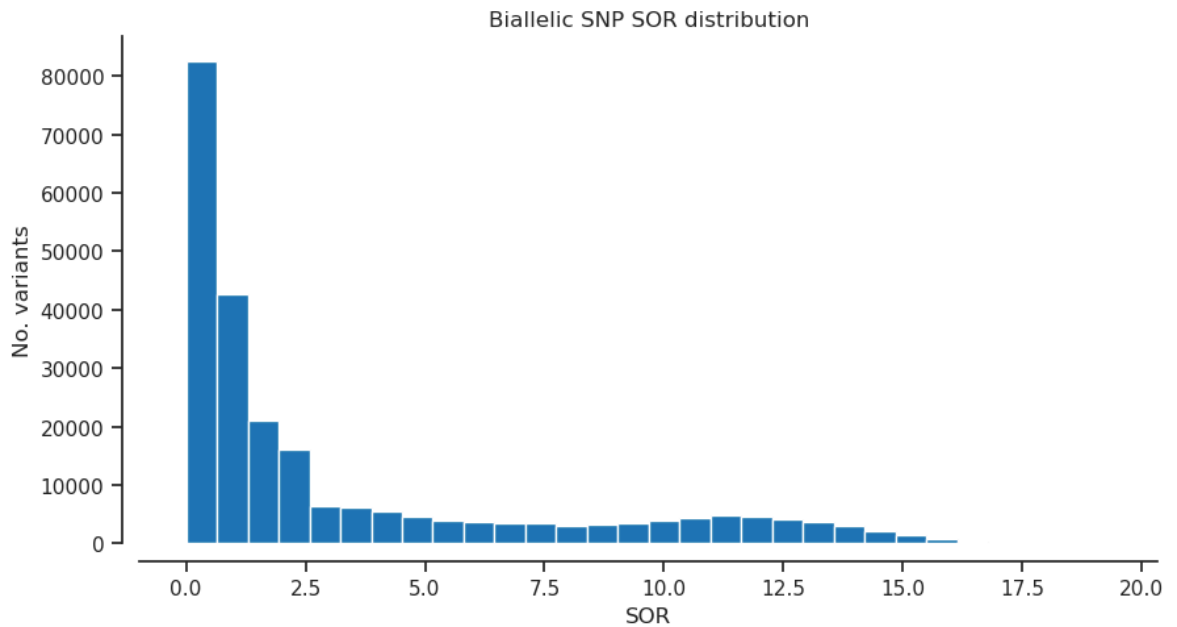


SOR - Symmetric Odds Ratio of 2x2 contingency table to detect strand bias

```
In [25]: plot_hist('SOR', 'var') # Symmetric Odds Ratio of 2x2 contingency table t
```

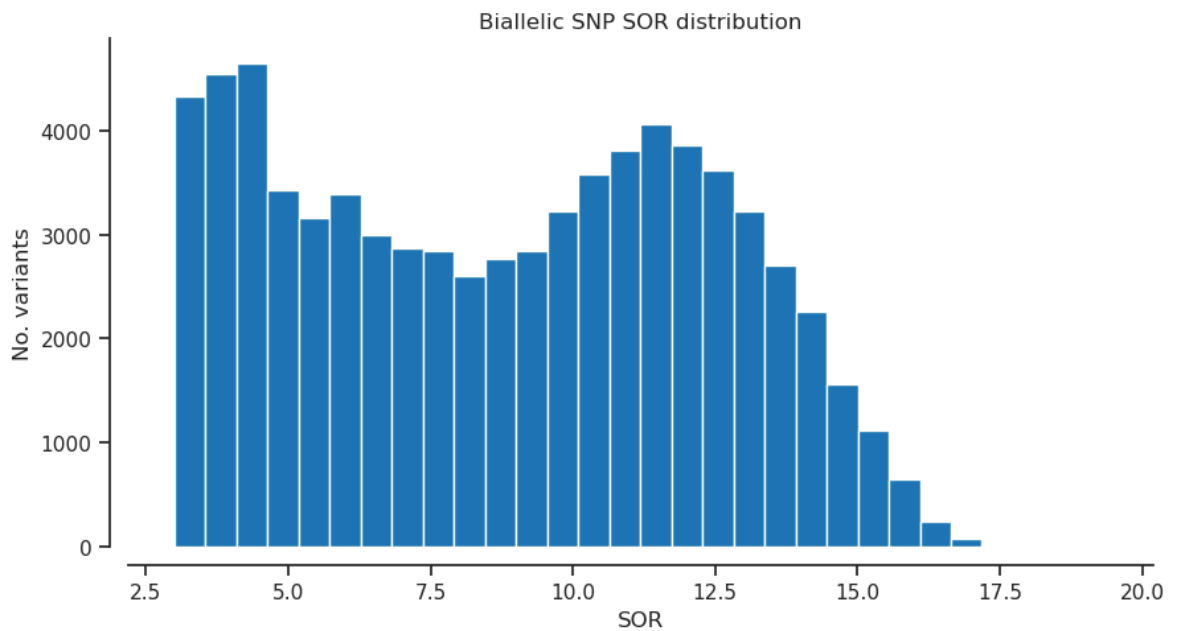


```
In [26]: plot_hist('SOR', 'biallelic') # Symmetric Odds Ratio of 2x2 contingency ta
```

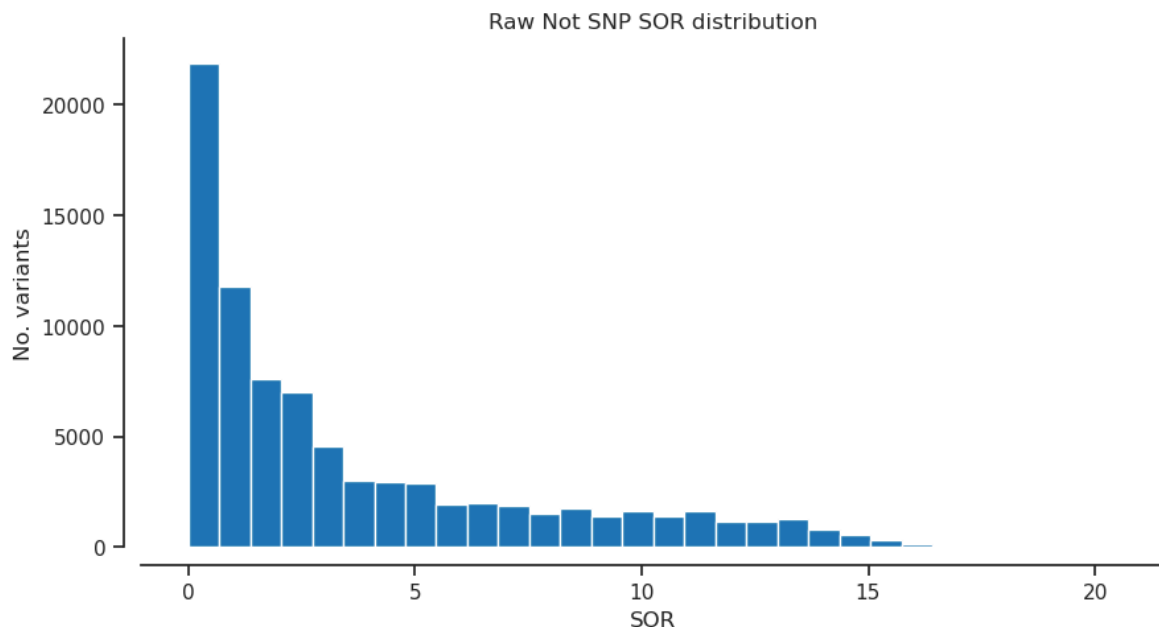


```
In [27]: filter_expression = '(SOR > 3)'
         bi_selection = biallelic_np.query(filter_expression)[:]
```

```
In [28]: plot_hist('SOR') # Symmetric Odds Ratio of 2x2 contingency table to detect
```

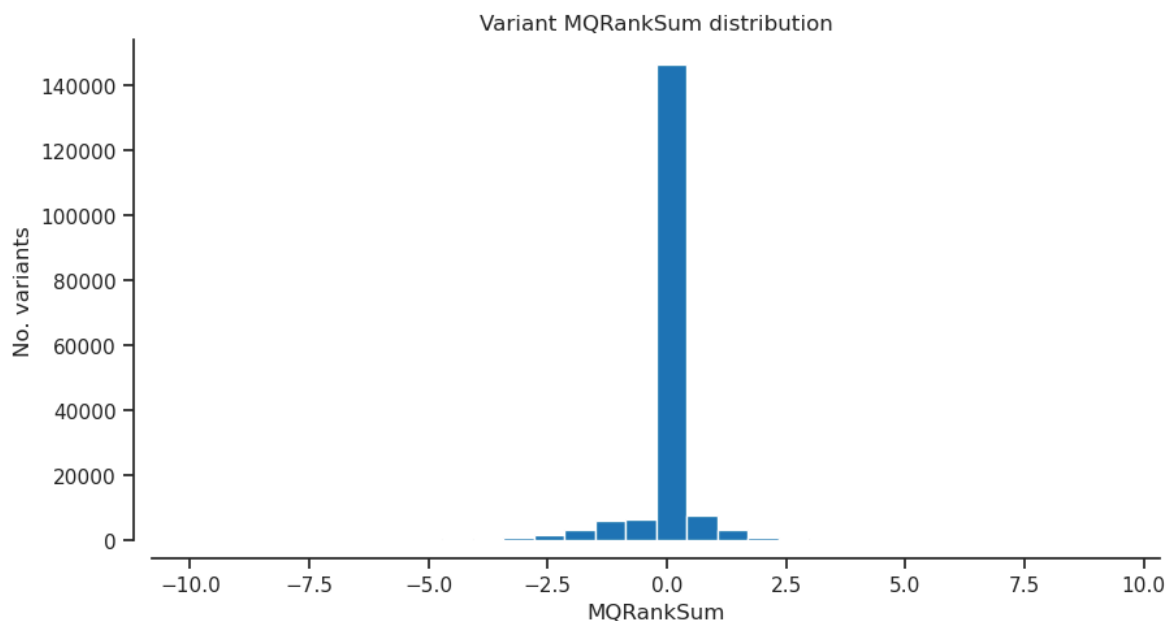


```
In [29]: plot_hist('SOR', 'notsnp') # Symmetric Odds Ratio of 2x2 contingency table
```

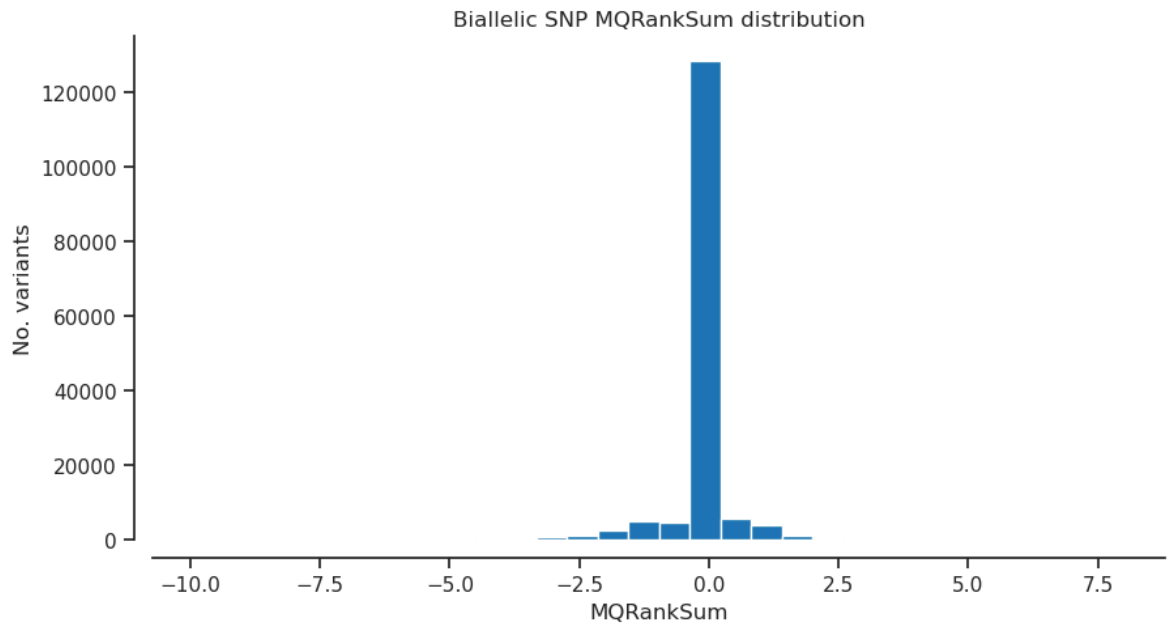


MQRankSum - Z-score From Wilcoxon rank sum test of Alt vs. Ref read mapping qualities

```
In [30]: plot_hist('MQRankSum','var') # Z-score From Wilcoxon rank sum test of Alt
```

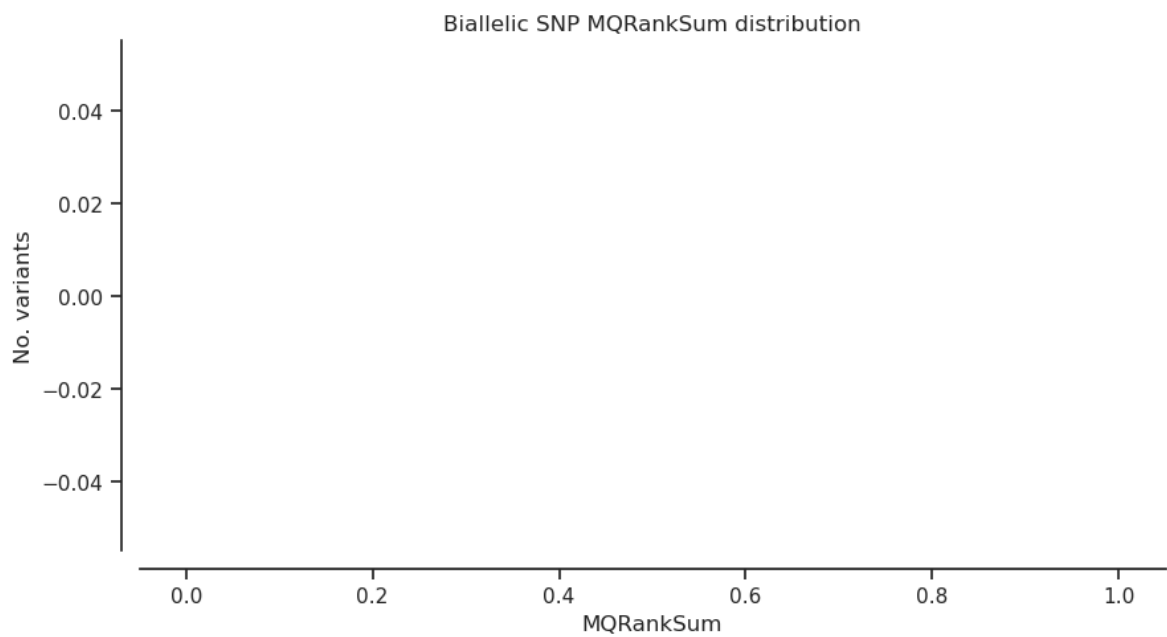


```
In [31]: plot_hist('MQRankSum','biallelic') # Z-score From Wilcoxon rank sum test
```

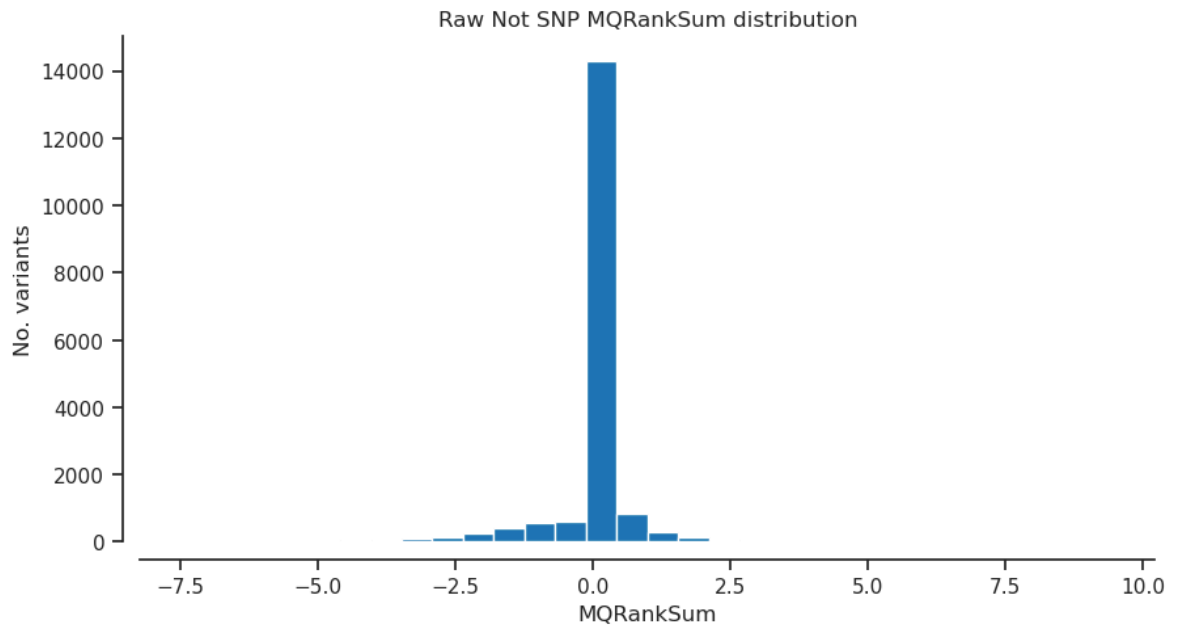


```
In [32]: filter_expression = '(MQRankSum < -12.5)'  
bi_selection = biallelic_np.query(filter_expression)[:]
```

```
In [33]: plot_hist('MQRankSum') # Z-score From Wilcoxon rank sum test of Alt vs. R
```

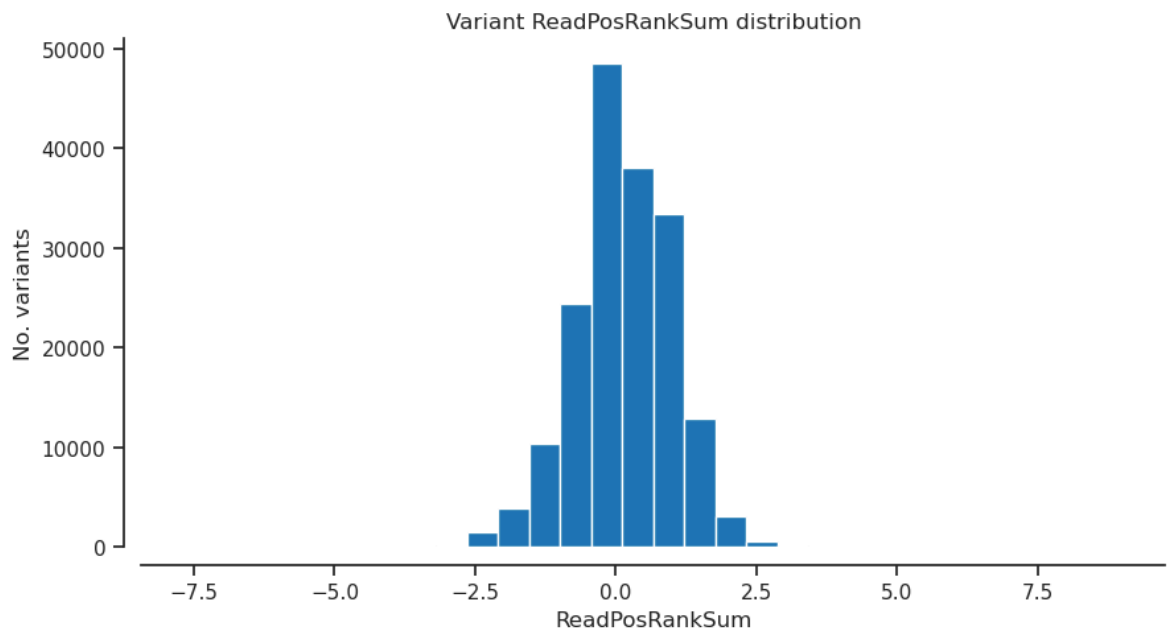


```
In [34]: plot_hist('MQRankSum', 'notsnp') # Z-score From Wilcoxon rank sum test of
```

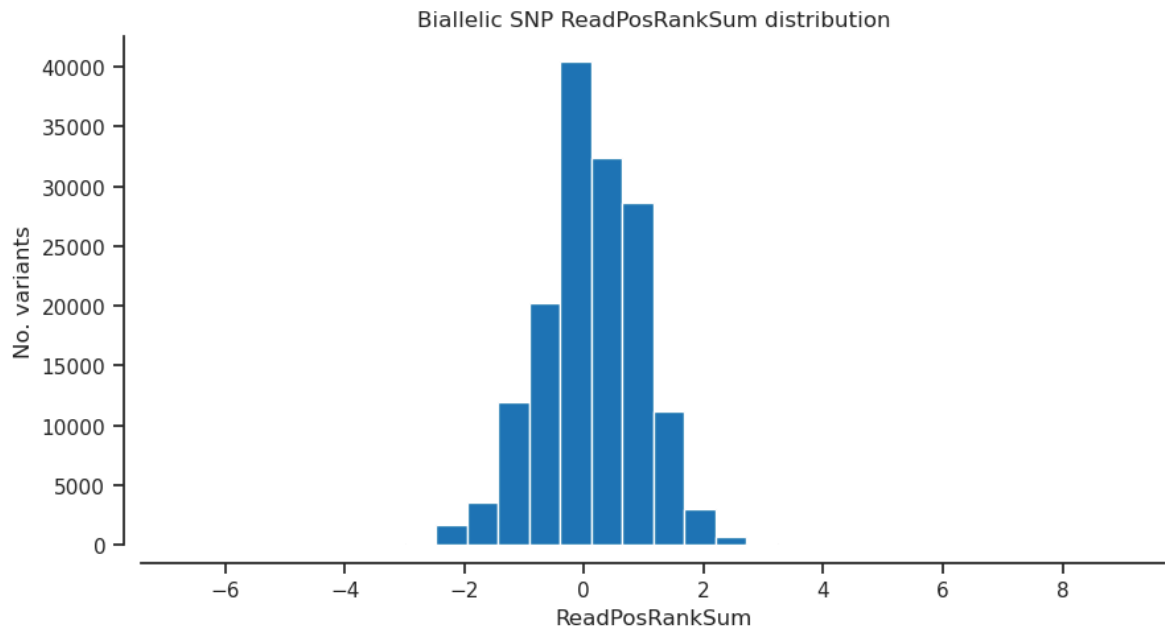


ReadPosRankSum - Z-score from Wilcoxon rank sum test of Alt vs. Ref read position bias

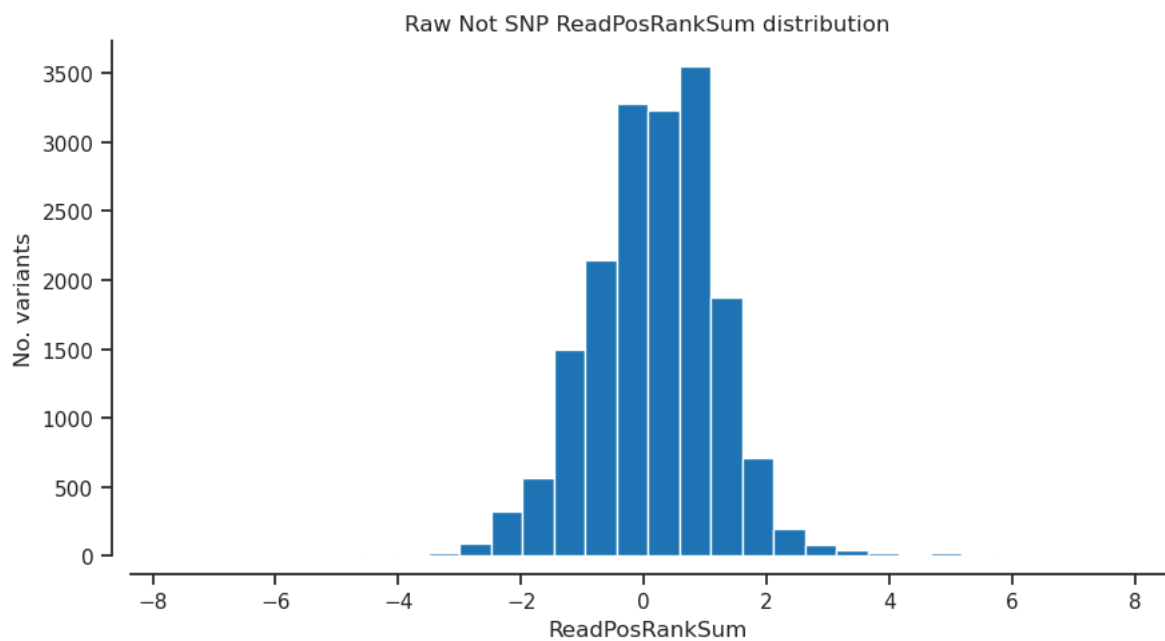
```
In [35]: plot_hist('ReadPosRankSum', 'var') # Z-score from Wilcoxon rank sum test o
```



```
In [36]: plot_hist('ReadPosRankSum', 'biallelic') # Z-score from Wilcoxon rank sum
```

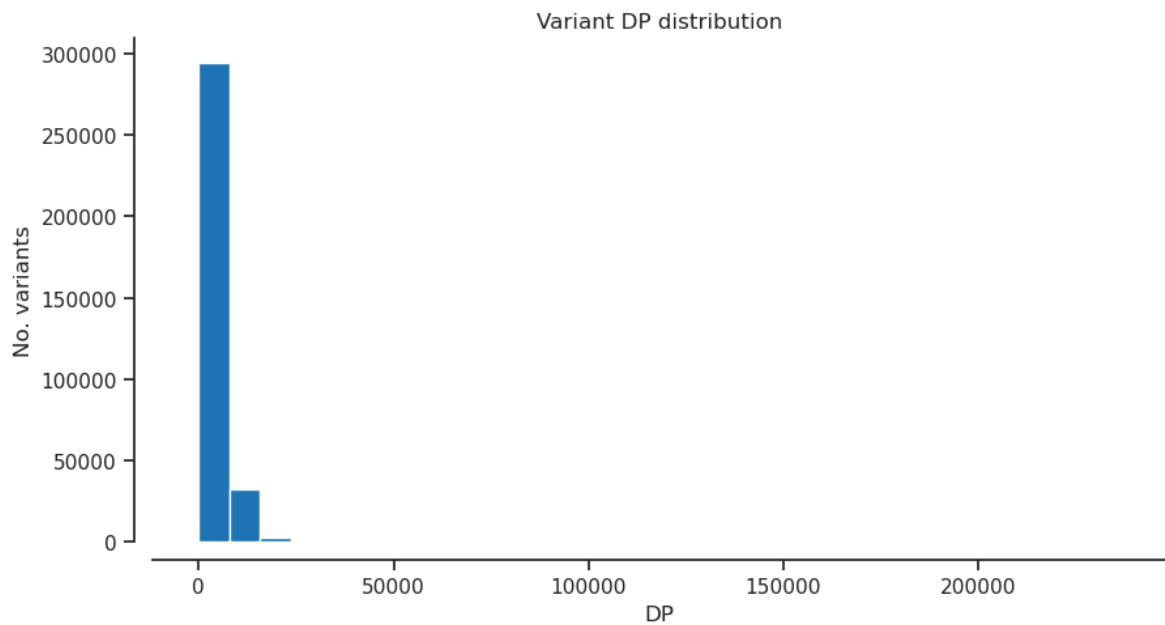



```
In [37]: plot_hist('ReadPosRankSum', 'notsnp') # Z-score from Wilcoxon rank sum tes
```

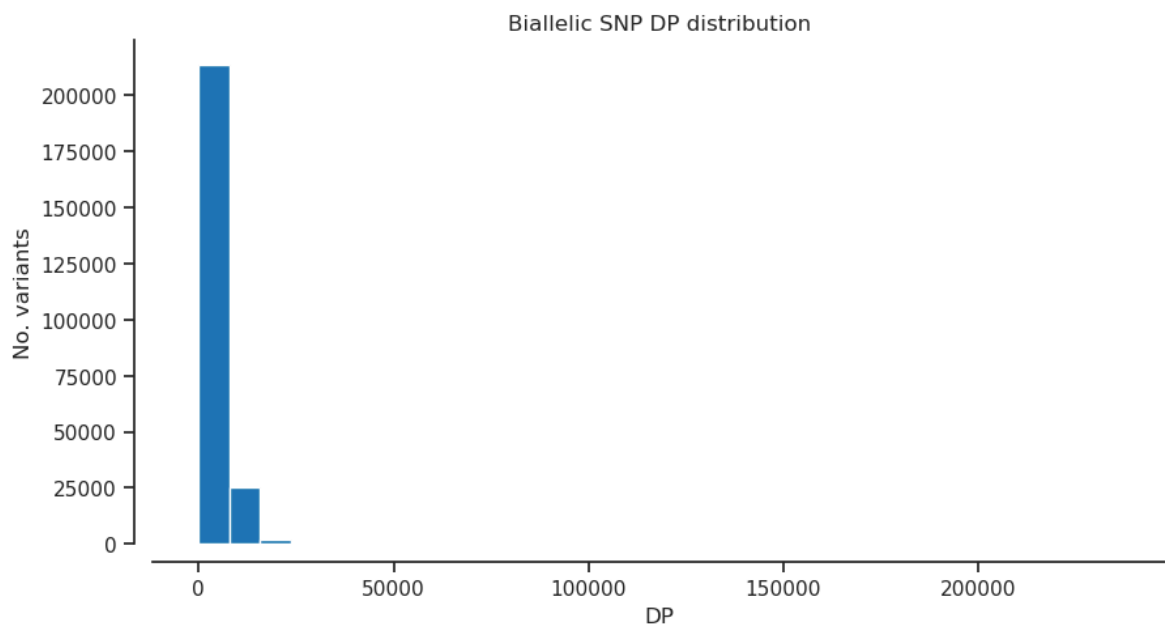


DP - Approximate read depth

```
In [38]: plot_hist('DP', 'var')
```

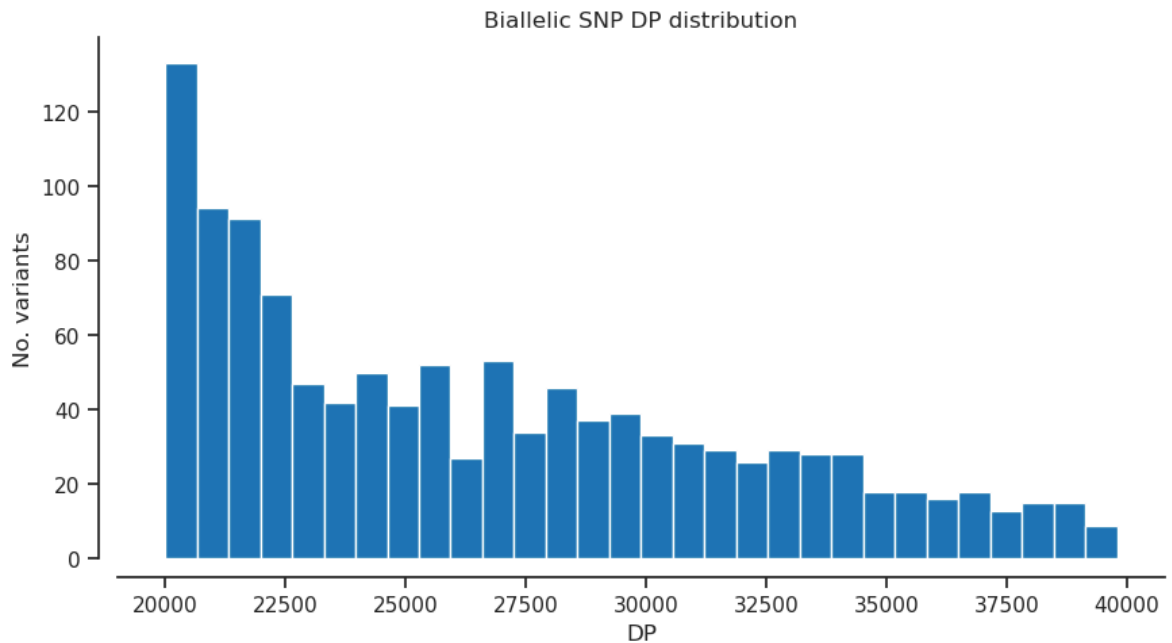


```
In [39]: plot_hist('DP', 'biallelic')
```

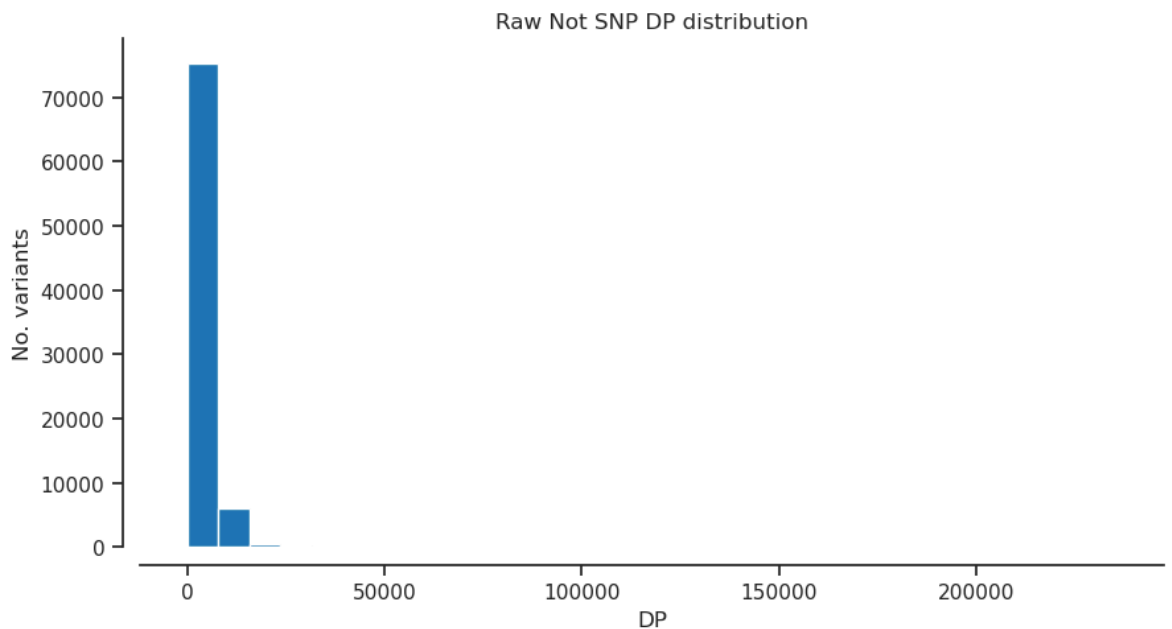


```
In [40]: filter_expression = '(DP > 20000) & (DP < 40000)'  
bi_selection = biallelic_np.query(filter_expression)[:]
```

```
In [41]: plot_hist('DP')
```

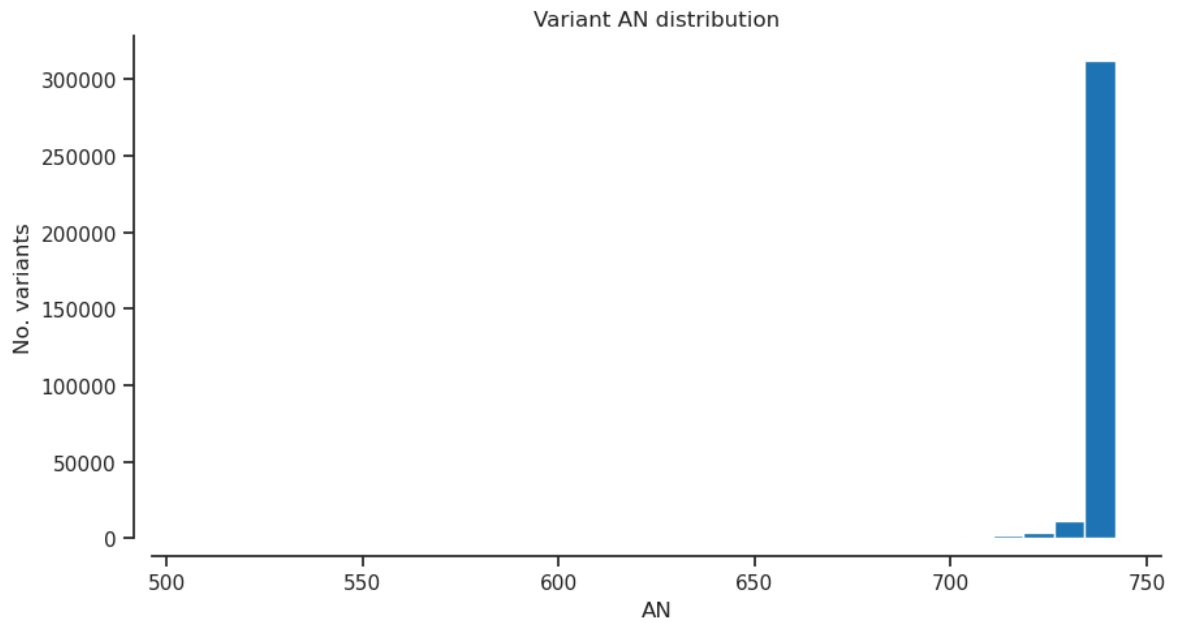


```
In [43]: plot_hist('DP', 'notsnr')
```

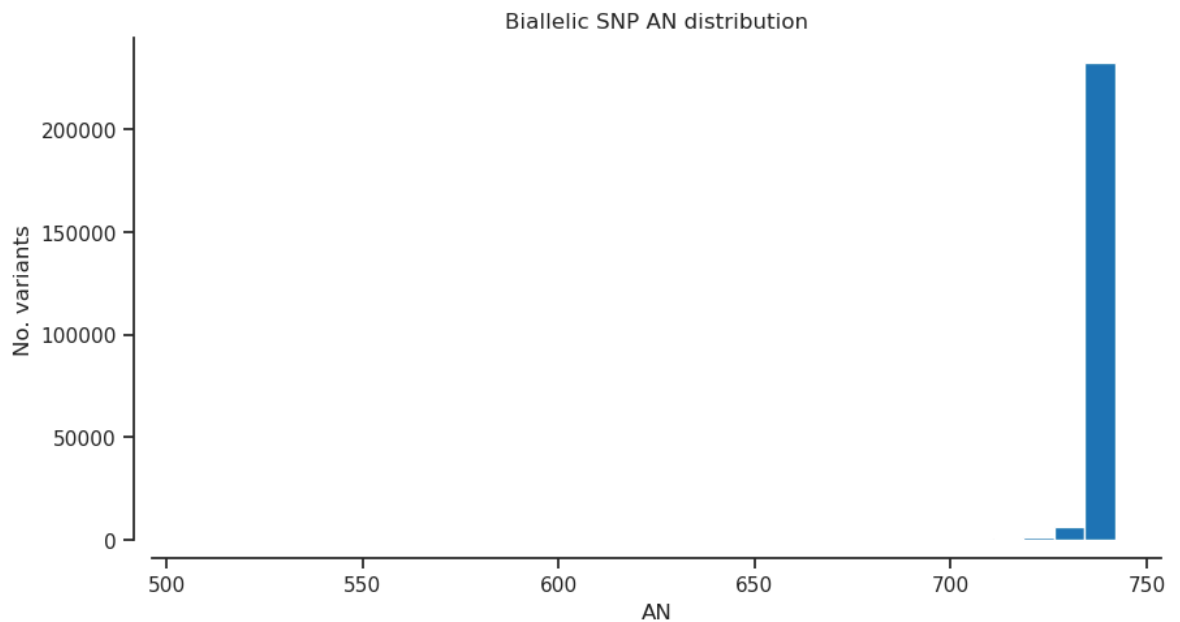


AN - Total number of alleles in called genotypes

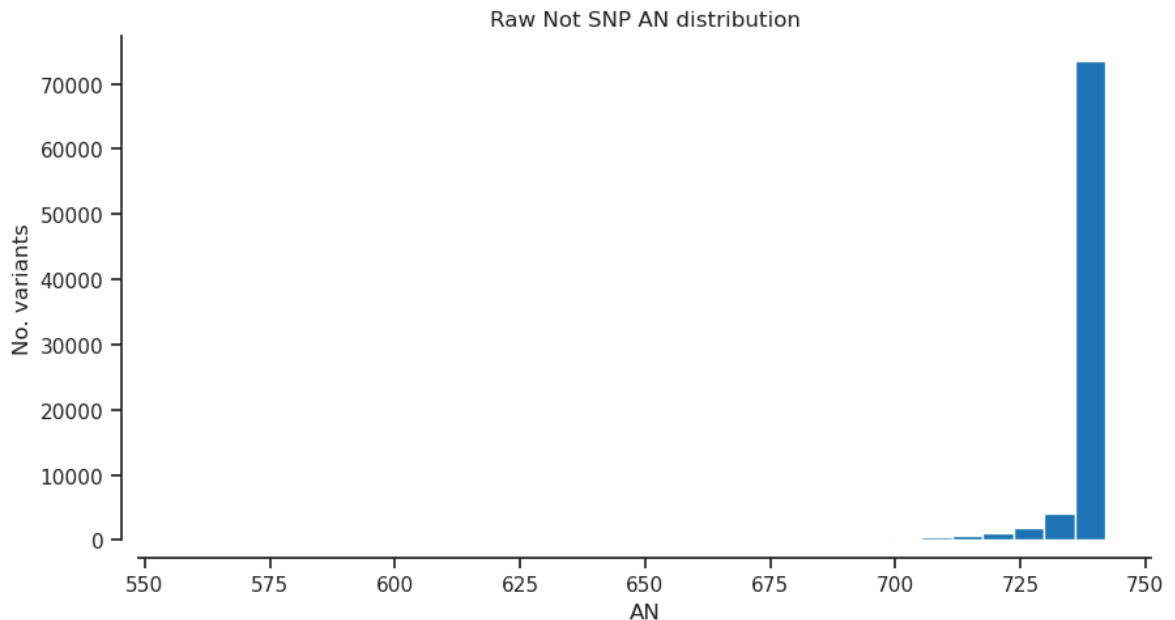
```
In [42]: plot_hist('AN', 'var') # Total number of alleles in called genotypes
```



```
In [43]: plot_hist('AN','biallelic') # Total number of alleles in called genotypes
```



```
In [44]: plot_hist('AN','notsnp') # Total number of alleles in called genotypes
```



Selected filter

```
In [45]: # QD: Variant Confidence/Quality by Depth
# AN: Total number of alleles in called genotypes
filter_expression = '(QD >= 2) & (MQ >= 40) & (MQRankSum >= -12.5) & (is_
variant_selection = variants_np.eval(filter_expression)[:])
np.count_nonzero(variant_selection)
```

Out[45]: 137640

Genotype

```
In [46]: calldata_var = callset_var['calldata']
list(calldata_var)
```

Out[46]: ['AD', 'DP', 'GQ', 'GT', 'MIN_DP', 'PGT', 'PID', 'PL', 'PS', 'RGQ', 'SB']

```
In [47]: genotypes_var = allele.GenotypeChunkedArray(calldata_var['GT'])
genotypes_var
```

Out [47]: <GenotypeChunkedArray shape=(331610, 371, 2) dtype=int8 chunks=(65536, 64, 2)
nbytes=234.7M cbytes=12.8M cratio=18.3 compression=gzip compression_opts=1
values=h5py._hl.dataset.Dataset>

	0	1	2	3	4	...	366	367	368	369	370
0	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
1	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
2	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
...	...										
331607	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
331608	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
331609	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0

```
In [48]: # using the selected filters set above  
gt_filtered_snps = genotypes_var.subset(variant_selection)  
gt_filtered_snps
```

Out [48]: <GenotypeChunkedArray shape=(137640, 371, 2) dtype=int8 chunks=(1076, 371, 2)
nbytes=97.4M cbytes=9.8M cratio=9.9 compression=blosc compression_opts=
{'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>

	0	1	2	3	4	...	366	367	368	369	370
0	1/1	1/1	1/1	1/1	1/1	...	1/1	1/1	1/1	1/1	1/1
1	0/0	0/0	0/1	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
2	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
...	...										
137637	1/1	1/1	1/1	1/1	1/1	...	1/1	0/0	0/0	1/1	1/1
137638	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
137639	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0

```
In [49]: # grab the allele counts for the populations  
ac = gt_filtered_snps.count_alleles()  
ac
```

Out [49]: <AlleleCountsChunkedArray shape=(137640, 4) dtype=int32 chunks=(17205, 4) nbytes=2.1M cbytes=431.6K cratio=5.0 compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>

	0	1	2	3
0	45	697	0	0
1	741	1	0	0
2	695	47	0	0
...	...			
137637	47	695	0	0
137638	741	1	0	0
137639	741	1	0	0

In [50]: `ac[:,]`

Out [50]: <AlleleCountsArray shape=(137640, 4) dtype=int32>

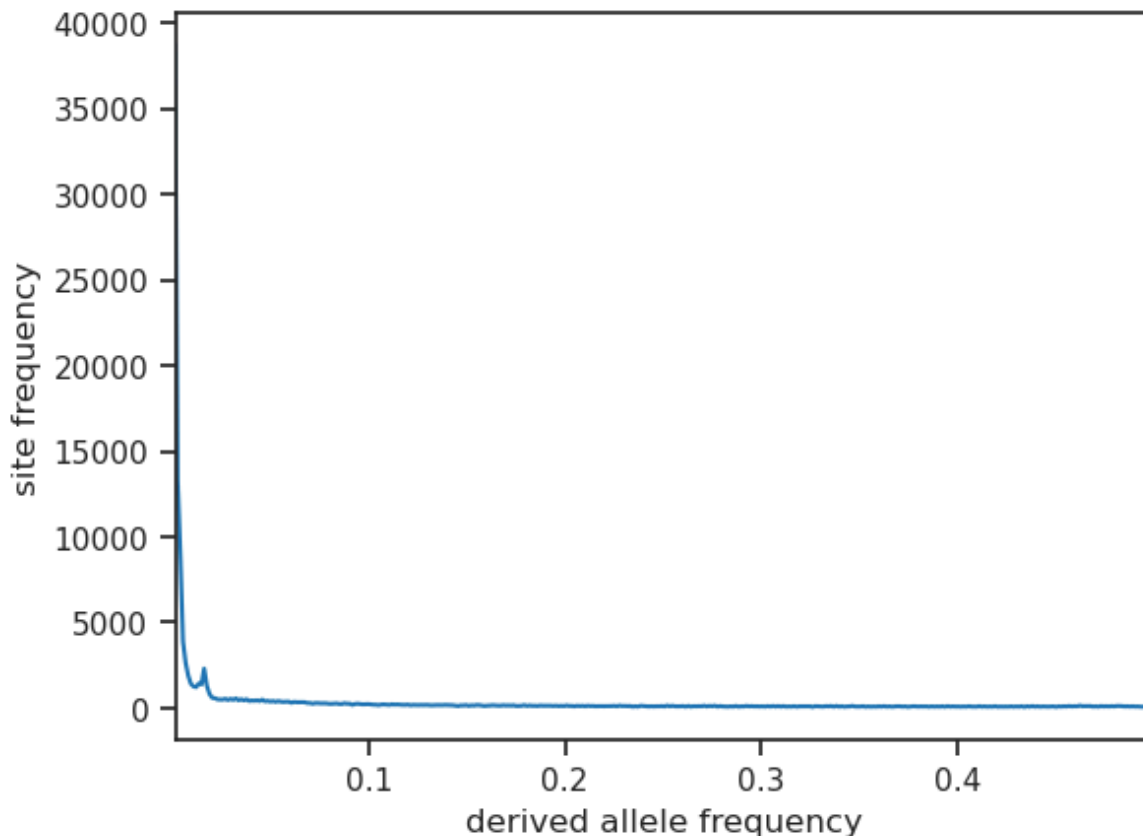
	0	1	2	3
0	45	697	0	0
1	741	1	0	0
2	695	47	0	0
...	...			
137637	47	695	0	0
137638	741	1	0	0
137639	741	1	0	0

In [51]: `# Which ones are biallelic?`
`is_biallelic_01 = ac.is_biallelic_01()[:,]`
`ac1 = ac.compress(is_biallelic_01, axis=0)[:, :2]`
`ac1`
##this part of the code is only for graphing the SFS, is not useful for f

Out [51]: `array([[45, 697],`
`[741, 1],`
`[695, 47],`
`...,`
`[47, 695],`
`[741, 1],`
`[741, 1]], dtype=int32)`

In [52]: `# plot the sfs of the derived allele`
`s = allel.sfs_folded(ac1)`
`allel.plot_sfs(s, yscale="linear", n=ac1.sum(axis=1).max())`

Out [52]: <Axes: xlabel='derived allele frequency', ylabel='site frequency'>



```
In [53]: biallelic = (ac.max_allele() == 1)
        ###This is the filter expression for biallelic sites
        biallelic
```

```
Out[53]: <ChunkedArrayWrapper shape=(137640,) dtype=bool chunks=(137640,)
        nbytes=134.4K cbytes=19.1K cratio=7.0
        compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shu
        ffile': 1, 'blocksize': 0}
        values=zarr.core.Array>
```

```
In [54]: # select only the biallelic variants
        gt_biallelic = gt_filtered_snps.compress(biallelic)
        gt_biallelic
```

```
Out[54]: <GenotypeChunkedArray shape=(132854, 371, 2) dtype=int8 chunks=(2076, 371, 2)
        nbytes=94.0M cbytes=9.2M cratio=10.2 compression=blosc compression_opts=
        {'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>
```

	0	1	2	3	4	...	366	367	368	369	370
0	1/1	1/1	1/1	1/1	1/1	...	1/1	1/1	1/1	1/1	1/1
1	0/0	0/0	0/1	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
2	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
...	...										
132851	1/1	1/1	1/1	1/1	1/1	...	1/1	0/0	0/0	1/1	1/1
132852	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
132853	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0


```
In [55]: n_variants = len(gt_biallelic)
         n_variants
```

```
Out[55]: 132854
```

```
In [56]: pc_missing = gt_biallelic.count_missing(axis=0)[:]* 100 / n_variants
         pc_het = gt_biallelic.count_het(axis=0)[:]* 100 / n_variants
```

Samples

```
In [57]: samples_var = callset_var['samples']
         samples_var = list(samples_var)
         samples_var
```

```
Out[57]: [b'ESP00057-001',  
          b'ESP00057-002',  
          b'ESP00057-003',  
          b'ESP00057-004',  
          b'ESP00057-005',  
          b'ESP00057-006',  
          b'ESP00057-007',  
          b'ESP00057-008',  
          b'ESP00057-009',  
          b'ESP00057-010',  
          b'ESP00057-011',  
          b'ESP00057-012',  
          b'ESP00057-013',  
          b'ESP00057-014',  
          b'ESP00057-015',  
          b'ESP00057-016',  
          b'ESP00057-017',  
          b'ESP00057-018',  
          b'ESP00057-019',  
          b'ESP00057-020',  
          b'ESP00057-021',  
          b'ESP00057-023',  
          b'ESP00057-024',  
          b'ESP00057-025',  
          b'ESP00091-001',  
          b'ESP00091-002',  
          b'ESP00091-003',  
          b'ESP00091-004',  
          b'ESP00091-005',  
          b'ESP00091-006',  
          b'ESP00091-007',  
          b'ESP00091-008',  
          b'ESP00091-009',  
          b'ESP00091-010',  
          b'ESP00091-011',  
          b'ESP00091-012',  
          b'ESP00091-013',  
          b'ESP00091-014',  
          b'ESP00091-015',  
          b'ESP00091-016',  
          b'ESP00091-017',  
          b'ESP00091-018',  
          b'ESP00091-019',  
          b'ESP00091-020',  
          b'ESP00091-022',  
          b'ESP00091-023',  
          b'ESP00091-024',  
          b'ESP00091-025',  
          b'ESP00094-001',  
          b'ESP00094-002',  
          b'ESP00094-003',  
          b'ESP00094-004',  
          b'ESP00094-005',  
          b'ESP00094-006',  
          b'ESP00094-007',  
          b'ESP00094-008',  
          b'ESP00094-009',  
          b'ESP00094-010',  
          b'ESP00094-011',  
          b'ESP00094-012',
```

b'ESP00094-013',
b'ESP00094-014',
b'ESP00094-015',
b'ESP00094-016',
b'ESP00094-017',
b'ESP00094-018',
b'ESP00094-019',
b'ESP00094-020',
b'ESP00094-021',
b'ESP00094-022',
b'ESP00094-023',
b'ESP00094-024',
b'ESP00106-001',
b'ESP00106-002',
b'ESP00106-003',
b'ESP00106-004',
b'ESP00106-005',
b'ESP00106-006',
b'ESP00106-007',
b'ESP00106-008',
b'ESP00106-009',
b'ESP00106-010',
b'ESP00106-011',
b'ESP00106-012',
b'ESP00106-013',
b'ESP00106-014',
b'ESP00106-015',
b'ESP00106-016',
b'ESP00106-017',
b'ESP00106-018',
b'ESP00106-019',
b'ESP00106-020',
b'ESP00106-021',
b'ESP00106-022',
b'ESP00106-023',
b'ESP00106-024',
b'ESP00106-025',
b'ESP00166-001',
b'ESP00166-002',
b'ESP00166-003',
b'ESP00166-004',
b'ESP00166-005',
b'ESP00166-006',
b'ESP00166-007',
b'ESP00166-008',
b'ESP00166-009',
b'ESP00166-010',
b'ESP00166-011',
b'ESP00166-012',
b'ESP00166-013',
b'ESP00166-014',
b'ESP00166-015',
b'ESP00166-016',
b'ESP00166-017',
b'ESP00166-018',
b'ESP00166-019',
b'ESP00166-020',
b'ESP00166-021',
b'ESP00166-022',
b'ESP00166-023',

b'ESP00166-024',
b'ESP00166-025',
b'ESP00377-001',
b'ESP00377-002',
b'ESP00377-003',
b'ESP00377-004',
b'ESP00377-005',
b'ESP00377-006',
b'ESP00377-007',
b'ESP00377-008',
b'ESP00377-009',
b'ESP00377-010',
b'ESP00377-011',
b'ESP00377-012',
b'ESP00377-013',
b'ESP00377-014',
b'ESP00377-015',
b'ESP00377-016',
b'ESP00377-017',
b'ESP00377-018',
b'ESP00377-019',
b'ESP00377-020',
b'ESP00377-021',
b'ESP00377-022',
b'ESP00377-023',
b'ESP00377-024',
b'ITA00025-001',
b'ITA00025-002',
b'ITA00025-003',
b'ITA00025-004',
b'ITA00025-005',
b'ITA00025-006',
b'ITA00025-007',
b'ITA00025-008',
b'ITA00025-009',
b'ITA00025-010',
b'ITA00025-011',
b'ITA00025-012',
b'ITA00025-013',
b'ITA00025-014',
b'ITA00025-015',
b'ITA00025-016',
b'ITA00025-017',
b'ITA00025-018',
b'ITA00025-019',
b'ITA00025-020',
b'ITA00025-021',
b'ITA00025-022',
b'ITA00025-023',
b'ITA00025-024',
b'ITA00025-025',
b'ITA00027-001',
b'ITA00027-002',
b'ITA00027-003',
b'ITA00027-004',
b'ITA00027-005',
b'ITA00027-006',
b'ITA00027-007',
b'ITA00027-008',
b'ITA00027-009',

b'ITA00027-010',
b'ITA00027-011',
b'ITA00027-012',
b'ITA00027-013',
b'ITA00027-014',
b'ITA00027-015',
b'ITA00027-016',
b'ITA00027-017',
b'ITA00027-018',
b'ITA00027-019',
b'ITA00027-020',
b'ITA00027-021',
b'ITA00027-022',
b'ITA00027-023',
b'ITA00027-024',
b'ITA00027-025',
b'ITA00046-001',
b'ITA00046-002',
b'ITA00046-003',
b'ITA00046-004',
b'ITA00046-005',
b'ITA00046-006',
b'ITA00046-007',
b'ITA00046-008',
b'ITA00046-009',
b'ITA00046-010',
b'ITA00046-011',
b'ITA00046-012',
b'ITA00046-013',
b'ITA00046-014',
b'ITA00046-015',
b'ITA00046-016',
b'ITA00046-017',
b'ITA00046-018',
b'ITA00046-019',
b'ITA00046-020',
b'ITA00046-021',
b'ITA00046-022',
b'ITA00046-023',
b'ITA00046-024',
b'ITA00046-025',
b'ITA00075-001',
b'ITA00075-002',
b'ITA00075-003',
b'ITA00075-004',
b'ITA00075-005',
b'ITA00075-006',
b'ITA00075-007',
b'ITA00075-008',
b'ITA00075-009',
b'ITA00075-010',
b'ITA00075-011',
b'ITA00075-012',
b'ITA00075-013',
b'ITA00075-014',
b'ITA00075-015',
b'ITA00075-016',
b'ITA00075-017',
b'ITA00075-018',
b'ITA00075-019',

b'ITA00075-020',
b'ITA00075-021',
b'ITA00075-022',
b'ITA00075-023',
b'ITA00075-024',
b'ITA00075-025',
b'ITA00076-001',
b'ITA00076-002',
b'ITA00076-003',
b'ITA00076-004',
b'ITA00076-005',
b'ITA00076-006',
b'ITA00076-007',
b'ITA00076-008',
b'ITA00076-009',
b'ITA00076-010',
b'ITA00076-011',
b'ITA00076-012',
b'ITA00076-013',
b'ITA00076-014',
b'ITA00076-015',
b'ITA00076-016',
b'ITA00076-017',
b'ITA00076-018',
b'ITA00076-019',
b'ITA00076-020',
b'ITA00076-021',
b'ITA00076-022',
b'ITA00076-023',
b'ITA00076-024',
b'ITA00076-025',
b'ITA00133-001',
b'ITA00133-002',
b'ITA00133-003',
b'ITA00133-004',
b'ITA00133-005',
b'ITA00133-006',
b'ITA00133-007',
b'ITA00133-008',
b'ITA00133-009',
b'ITA00133-010',
b'ITA00133-011',
b'ITA00133-012',
b'ITA00133-013',
b'ITA00133-014',
b'ITA00133-015',
b'ITA00133-016',
b'ITA00133-017',
b'ITA00133-018',
b'ITA00133-019',
b'ITA00133-020',
b'ITA00133-021',
b'ITA00133-022',
b'ITA00133-023',
b'ITA00133-024',
b'ITA00133-025',
b'ITA00165-001',
b'ITA00165-002',
b'ITA00165-003',
b'ITA00165-004',

b' ITA00165-005',
b' ITA00165-006',
b' ITA00165-007',
b' ITA00165-008',
b' ITA00165-009',
b' ITA00165-010',
b' ITA00165-011',
b' ITA00165-012',
b' ITA00165-013',
b' ITA00165-014',
b' ITA00165-015',
b' ITA00165-016',
b' ITA00165-017',
b' ITA00165-018',
b' ITA00165-019',
b' ITA00165-020',
b' ITA00165-021',
b' ITA00165-022',
b' ITA00165-023',
b' ITA00165-024',
b' ITA00165-025',
b' ITA00261-001',
b' ITA00261-002',
b' ITA00261-003',
b' ITA00261-004',
b' ITA00261-005',
b' ITA00261-006',
b' ITA00261-007',
b' ITA00261-008',
b' ITA00261-009',
b' ITA00261-010',
b' ITA00261-011',
b' ITA00261-012',
b' ITA00261-013',
b' ITA00261-014',
b' ITA00261-015',
b' ITA00261-016',
b' ITA00261-017',
b' ITA00261-018',
b' ITA00261-019',
b' ITA00261-020',
b' ITA00261-021',
b' ITA00261-022',
b' ITA00261-023',
b' ITA00261-024',
b' ITA00261-025',
b' ITA00270-001',
b' ITA00270-002',
b' ITA00270-003',
b' ITA00270-004',
b' ITA00270-005',
b' ITA00270-006',
b' ITA00270-007',
b' ITA00270-008',
b' ITA00270-009',
b' ITA00270-010',
b' ITA00270-011',
b' ITA00270-012',
b' ITA00270-013',
b' ITA00270-014',

```

b'ITA00270-015',
b'ITA00270-016',
b'ITA00270-017',
b'ITA00270-018',
b'ITA00270-019',
b'ITA00270-020',
b'ITA00270-021',
b'ITA00270-022',
b'ITA00270-023',
b'ITA00270-024',
b'ITA00270-025']

```

```

In [58]: samples_fn = '~/scratch/data/Phalepensis/Pinus_halepensis_sample_list_sci
samples = pandas.read_csv(samples_fn, sep='\t')
samples

```

Out [58]:

	ID	Population
0	ESP00057-001	ESP00057
1	ESP00057-002	ESP00057
2	ESP00057-003	ESP00057
3	ESP00057-004	ESP00057
4	ESP00057-005	ESP00057
...
366	ITA00270-021	ITA00270
367	ITA00270-022	ITA00270
368	ITA00270-023	ITA00270
369	ITA00270-024	ITA00270
370	ITA00270-025	ITA00270

371 rows × 2 columns

```

In [59]: samples.Population.value_counts()

```

Out [59]:

Population	
ITA00025	25
ESP00166	25
ESP00106	25
ITA00076	25
ITA00075	25
ITA00046	25
ITA00027	25
ITA00133	25
ITA00165	25
ITA00261	25
ITA00270	25
ESP00377	24
ESP00057	24
ESP00091	24
ESP00094	24

Name: count, dtype: int64


```
In [60]: populations = samples.Population.unique()
populations
###This identifiers come from the metadata file
```

```
Out[60]: array(['ESP00057', 'ESP00091', 'ESP00094', 'ESP00106', 'ESP00166',
               'ESP00377', 'ITA00025', 'ITA00027', 'ITA00046', 'ITA00075',
               'ITA00076', 'ITA00133', 'ITA00165', 'ITA00261', 'ITA00270'],
          dtype=object)
```

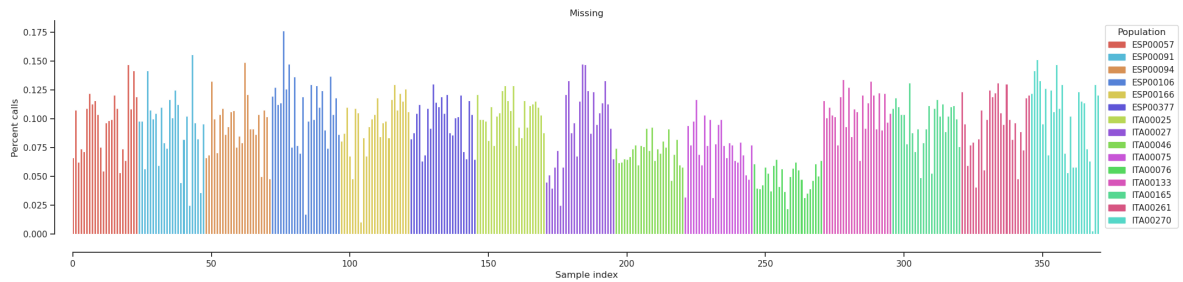
Gt frequency function

```
In [115... def plot_genotype_frequency(pc, title):
    fig, ax = plt.subplots(figsize=(24, 5))
    sns.despine(ax=ax, offset=24)
    left = np.arange(len(pc))
    palette = sns.color_palette("hls", 15)
    pop2color = {'ESP00057': palette[0],
                  'ESP00091': palette[8],
                  'ESP00094': palette[1],
                  'ESP00106': palette[9],
                  'ESP00166': palette[2],
                  'ESP00377': palette[10],
                  'ITA00025': palette[3],
                  'ITA00027': palette[11],
                  'ITA00046': palette[4],
                  'ITA00075': palette[12],
                  'ITA00076': palette[5],
                  'ITA00133': palette[13],
                  'ITA00165': palette[6],
                  'ITA00261': palette[14],
                  'ITA00270': palette[7]}

    colors = [pop2color[p] for p in samples.Population]
    ax.bar(left, pc, color=colors)
    ax.set_xlim(0, len(pc))
    ax.set_xlabel('Sample index')
    ax.set_ylabel('Percent calls')
    ax.set_title(title)
    handles = [mpl.patches.Patch(color=palette[0]),
                mpl.patches.Patch(color=palette[8]),
                mpl.patches.Patch(color=palette[1]),
                mpl.patches.Patch(color=palette[9]),
                mpl.patches.Patch(color=palette[2]),
                mpl.patches.Patch(color=palette[10]),
                mpl.patches.Patch(color=palette[3]),
                mpl.patches.Patch(color=palette[11]),
                mpl.patches.Patch(color=palette[4]),
                mpl.patches.Patch(color=palette[12]),
                mpl.patches.Patch(color=palette[5]),
                mpl.patches.Patch(color=palette[13]),
                mpl.patches.Patch(color=palette[6]),
                mpl.patches.Patch(color=palette[14]),
                mpl.patches.Patch(color=palette[7]),
                mpl.patches.Patch(color="#DFFF00")]
    ax.legend(handles=handles, labels=['ESP00057', 'ESP00091', 'ESP00094',
                                      'ESP00377', 'ITA00025', 'ITA00027', 'ITA00046', 'ITA00075',
                                      'ITA00076', 'ITA00133', 'ITA00165', 'ITA00261', 'ITA00270'], title
              bbox_to_anchor=(1, 1), loc='upper left')
```

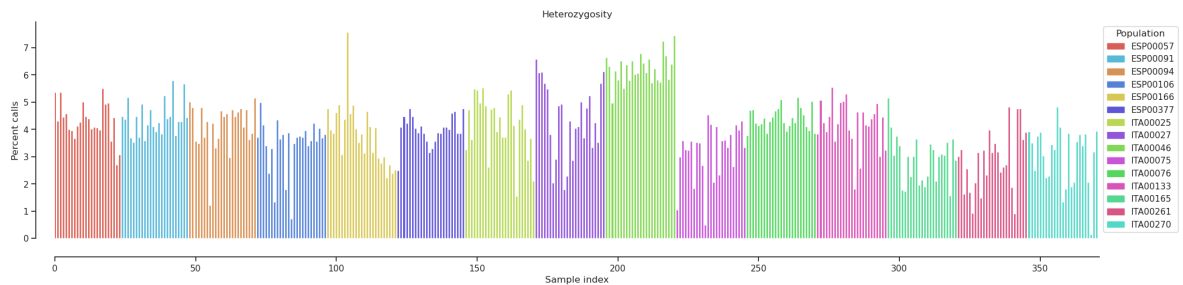
Plot missing

```
In [116... plot_genotype_frequency(pc_missing, 'Missing')
```



Plot heterozygosity

```
In [117... plot_genotype_frequency(pc_het, 'Heterozygosity')
```



PCA

```
In [113... palette = sns.color_palette("hls",15)
pop_colours = {
    'ESP00057': palette[0],
    'ESP00091': palette[8],
    'ESP00094': palette[1],
    'ESP00106': palette[9],
    'ESP00166': palette[2],
    'ESP00377': palette[10],
    'ITA00025': palette[3],
    'ITA00027': palette[11],
    'ITA00046': palette[4],
    'ITA00075': palette[12],
    'ITA00076': palette[5],
    'ITA00133': palette[13],
    'ITA00165': palette[6],
    'ITA00261': palette[14],
    'ITA00270': palette[7]
}
```

```
In [111... def plot_pca_coords(coords, model, pc1, pc2, ax, sample_population):
    sns.despine(ax=ax, offset=5)
    x = coords[:, pc1]
    y = coords[:, pc2]
    for pop in populations:
        flt = (sample_population == pop)
        ax.plot(x[flt], y[flt], marker='o', linestyle='', color=pop_color,
                label=pop, markersize=6, mec='k', mew=.5)
```

```
ax.set_xlabel('PC%s (%.1f%%)' % (pc1+1, model.explained_variance_ratio_))
ax.set_ylabel('PC%s (%.1f%%)' % (pc2+1, model.explained_variance_ratio_))
```

```
def fig_pca(coords, model, title, sample_population=None):
    if sample_population is None:
        sample_population = samples.Population
    # plot coords for PCs 1 vs 2, 3 vs 4
    fig = plt.figure(figsize=(10, 5))
    ax = fig.add_subplot(1, 2, 1)
    plot_pca_coords(coords, model, 0, 1, ax, sample_population)
    ax = fig.add_subplot(1, 2, 2)
    plot_pca_coords(coords, model, 2, 3, ax, sample_population)
    ax.legend(bbox_to_anchor=(1, 1), loc='upper left')
    fig.suptitle(title, y=1.02)
    fig.tight_layout()
```

```
In [99]: ac2 = gt_biallelic.count_alleles()
ac2
```

```
Out[99]: <AlleleCountsChunkedArray shape=(132854, 2) dtype=int32 chunks=(33214, 2)
nbytes=1.0M cbytes=315.0K cratio=3.3 compression=blosc compression_opts=
{'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>
```

	0	1
0	45	697
1	741	1
2	695	47
...
132851	47	695
132852	741	1
132853	741	1

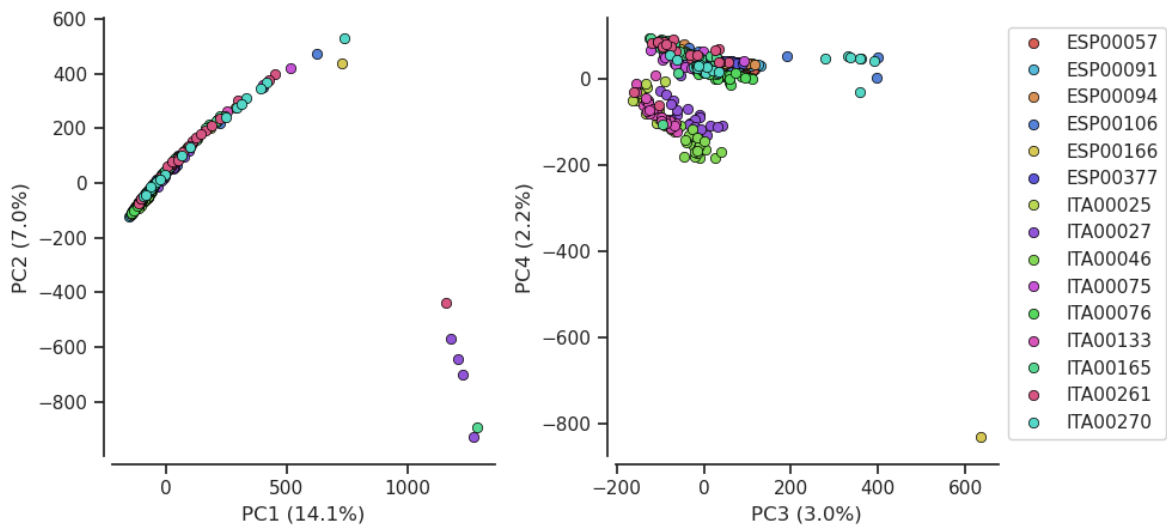
```
In [100... flt = (ac2[:, :2].min(axis=1) > 1)
gf = gt_biallelic.compress(flt, axis=0)
gn = gf.to_n_alt()
gn
```

```
Out[100... <ChunkedArrayWrapper shape=(94134, 371) dtype=int8 chunks=(2942, 371)
nbytes=33.3M cbytes=6.2M cratio=5.3
compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0}
values=zarr.core.Array>
```

```
In [101... coords1, model1 = allel.pca(gn, n_components=10, scaler='patterson')
```

```
In [114... fig_pca(coords1, model1, 'Figure 1. Conventional PCA.')
```

Figure 1. Conventional PCA.



```
In [103... weirdos = coords1[:,2]>600
samples[weirdos]
```

Out[103...

	ID	Population
104	ESP00166-008	ESP00166

```
In [104... pc_het[weirdos]
```

Out[104... array([7.57448026])

```
In [105... pc_missing[weirdos]
```

Out[105... array([0.01053788])

```
In [106... weirdos2 = coords1[:,1]<-300
samples[weirdos2]
```

Out[106...

	ID	Population
176	ITA00027-006	ITA00027
177	ITA00027-007	ITA00027
181	ITA00027-011	ITA00027
182	ITA00027-012	ITA00027
320	ITA00165-025	ITA00165
323	ITA00261-003	ITA00261

```
In [74]: weirdos3 = coords1[:,1]<-800
samples[weirdos3]
```

Out [74]:

	ID	Population
176	ITA00027-006	ITA00027
320	ITA00165-025	ITA00165

In [75]: pc_het[weirdos2]

Out [75]: array([3.81245578, 2.03381155, 1.79068752, 2.29801135, 2.86479895, 1.63412468])

In [76]: pc_missing[weirdos2]

Out [76]: array([0.0248393 , 0.05795836, 0.09634637, 0.06774354, 0.0760233 , 0.05946377])

In [77]: pc_het[weirdos3]

Out [77]: array([3.81245578, 2.86479895])

In [78]: pc_missing[weirdos3]

Out [78]: array([0.0248393, 0.0760233])

In [79]: coords1[323]

Out [79]: array([1156.3466 , -438.864 , -27.97357 , 46.599735, 6.090253, 170.64516 , 76.420944, 742.7226 , -94.13305 , 430.47336], dtype=float32)

In [80]: coords1[320]

Out [80]: array([1287.2603 , -890.67163 , -59.01289 , 11.353212, -18.697811, -118.72887 , -18.152779, -339.92252 , 76.887115, 354.73297], dtype=float32)

In [81]: coords1[182]

Out [81]: array([1224.7372 , -700.68964 , -52.97877 , 25.49761 , -10.2087 , -18.444567, -14.932702, -24.474918, -19.220716, -269.9098], dtype=float32)