In [1]:
```
import sys
!{sys.executable} -m pip install --user scikit-allel
```

```
Requirement already satisfied: scikit-allel in /projappl/project_2009301/s
oftware/lib/python3.10/site-packages (1.3.8)
Requirement already satisfied: numpy in /PUHTI_TYKKY_FRQGCcR/miniconda/env
s/env1/lib/python3.10/site-packages (from scikit-allel) (1.26.4)
Requirement already satisfied: dask[array] in /PUHTI_TYKKY_FRQGCcR/minicon
da/envs/env1/lib/python3.10/site-packages (from scikit-allel) (2024.4.1)
Requirement already satisfied: click>=8.1 in /PUHTI_TYKKY_FRQGCcR/minicond
a/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit-allel)
(8.1.7)
Requirement already satisfied: cloudpickle>=1.5.0 in /PUHTI_TYKKY_FRQGCcR/
miniconda/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit
-allel) (3.0.0)
Requirement already satisfied: fsspec>=2021.09.0 in /PUHTI_TYKKY_FRQGCcR/m
iniconda/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit-
allel) (2024.3.1)
Requirement already satisfied: packaging>=20.0 in /PUHTI_TYKKY_FRQGCcR/min
iconda/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit-al
lel) (23.2)
Requirement already satisfied: partd>=1.2.0 in /PUHTI_TYKKY_FRQGCcR/minico
nda/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit-alle
l) (1.4.1)
Requirement already satisfied: pyyaml>=5.3.1 in /PUHTI_TYKKY_FRQGCcR/minic
onda/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit-alle
l) (6.0.1)
Requirement already satisfied: toolz>=0.10.0 in /PUHTI_TYKKY_FRQGCcR/minic
onda/envs/env1/lib/python3.10/site-packages (from dask[array]->scikit-alle
l) (0.12.1)
Requirement already satisfied: importlib-metadata>=4.13.0 in /PUHTI_TYKKY_
FRQGCcR/miniconda/envs/env1/lib/python3.10/site-packages (from dask[array]
->scikit-allel) (7.1.0)
Requirement already satisfied: zipp>=0.5 in /PUHTI_TYKKY_FRQGCcR/minicond
a/envs/env1/lib/python3.10/site-packages (from importlib-metadata>=4.13.0-
>dask[array]->scikit-allel) (3.17.0)
Requirement already satisfied: locket in /PUHTI_TYKKY_FRQGCcR/miniconda/en
vs/env1/lib/python3.10/site-packages (from partd>=1.2.0->dask[array]->scik
it-allel) (1.0.0)
```

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/Anebrodensis/vcf_filter
```

Get data

In [3]:
```
callset_var_fn = '/users/mcevoysu/scratch/output/Anebrodensis/scikit-alle
callset_var = h5py.File(callset_var_fn, mode='r')
```

In [4]:
```
calldata_var = callset_var['calldata']
list(calldata_var)
```

Out[4]:
```
['AD', 'DP', 'GQ', 'GT', 'MIN_DP', 'PGT', 'PID', 'PL', 'PS', 'RGQ', 'S
B']
```

In [5]:
```
list(callset_var['variants'])
```

Out[5]:
```
['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 [6]:
```
variants = allel.VariantChunkedTable(callset_var['variants'])
variants
```

Out[6]: <VariantChunkedTable shape=(74891,) 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=12.8M cbytes=2.6M cratio=4.9 values=h5py._hl.group.Group>

	AC	AF	ALT	AN	BaseQRankSum	CHROM	DP	END	Ex
0	[4 -1 -1]	[0.286 nan nan]	[b'T' b'' b'']	14	1.11	b'aalba5_s00000080'	96	-1	
1	[4 -1 -1]	[0.286 nan nan]	[b'T' b'' b'']	14	0.0	b'aalba5_s00000080'	36	-1	
2	[4 -1 -1]	[0.286 nan nan]	[b'G' b'' b'']	14	0.0	b'aalba5_s00000080'	24	-1	
...									
74888	[6 -1 -1]	[0.1 nan nan]	[b'C' b'' b'']	60	0.0	b'aalba5_s01418300'	156	-1	
74889	[1 -1 -1]	[0.017 nan nan]	[b'A' b'' b'']	60	0.0	b'aalba5_s01418300'	117	-1	
74890	[1 -1 -1]	[0.017 nan nan]	[b'C' b'' b'']	60	0.765	b'aalba5_s01418300'	80	-1	

In [7]:
```
variants_np = variants[:]
rawsnps = variants_np.query('(is_snp == True)')
rawsnps
```

Out[7]: <VariantTable shape=(50778,) 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	Ex
0	[4 -1 -1]	[0.286 nan nan]	[b'T' b'' b'']	14	1.11	b'aalba5_s00000080'	96	-1	
1	[4 -1 -1]	[0.286 nan nan]	[b'T' b'' b'']	14	0.0	b'aalba5_s00000080'	36	-1	
2	[4 -1 -1]	[0.286 nan nan]	[b'G' b'' b'']	14	0.0	b'aalba5_s00000080'	24	-1	
...									
50775	[6 -1 -1]	[0.1 nan nan]	[b'C' b'' b'']	60	0.0	b'aalba5_s01418300'	156	-1	
50776	[1 -1 -1]	[0.017 nan nan]	[b'A' b'' b'']	60	0.0	b'aalba5_s01418300'	117	-1	
50777	[1 -1 -1]	[0.017 nan nan]	[b'C' b'' b'']	60	0.765	b'aalba5_s01418300'	80	-1	

In [8]:
```
notsnp = variants_np.query('(is_snp != True)')
notsnp
```

Out[8]: <VariantTable shape=(24113,) 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	Exc
0	[1 -1 -1]	[0.023 nan nan]	[b'*' b'' b'']	44	nan	b'aalba5_s00000180'	173	-1	
1	[2 -1 -1]	[0.04 nan nan]	[b'*' b'' b'']	50	nan	b'aalba5_s00000198'	331	-1	0
2	[2 -1 -1]	[0.04 nan nan]	[b'*' b'' b'']	50	nan	b'aalba5_s00000198'	330	-1	0
...									
24110	[5 -1 -1]	[0.083 nan nan]	[b'*' b'' b'']	60	nan	b'aalba5_s01418300'	61	-1	
24111	[5 -1 -1]	[0.083 nan nan]	[b'*' b'' b'']	60	nan	b'aalba5_s01418300'	61	-1	
24112	[5 -1 -1]	[0.083 nan nan]	[b'*' b'' b'']	60	nan	b'aalba5_s01418300'	61	-1	

Plot function

In [9]:
```
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 [10]:
```
numalt = rawsnps['numalt']
np.max(numalt)
```

Out[10]:
```
3
```

In [11]:
```
count_numalt = np.bincount(numalt)
count_numalt
```

Out[11]:
```
array([    0, 50252,   523,     3])
```

In [12]:
```
n_multiallelic = np.sum(count_numalt[2:])
n_multiallelic
```

Out[12]:
```
526
```

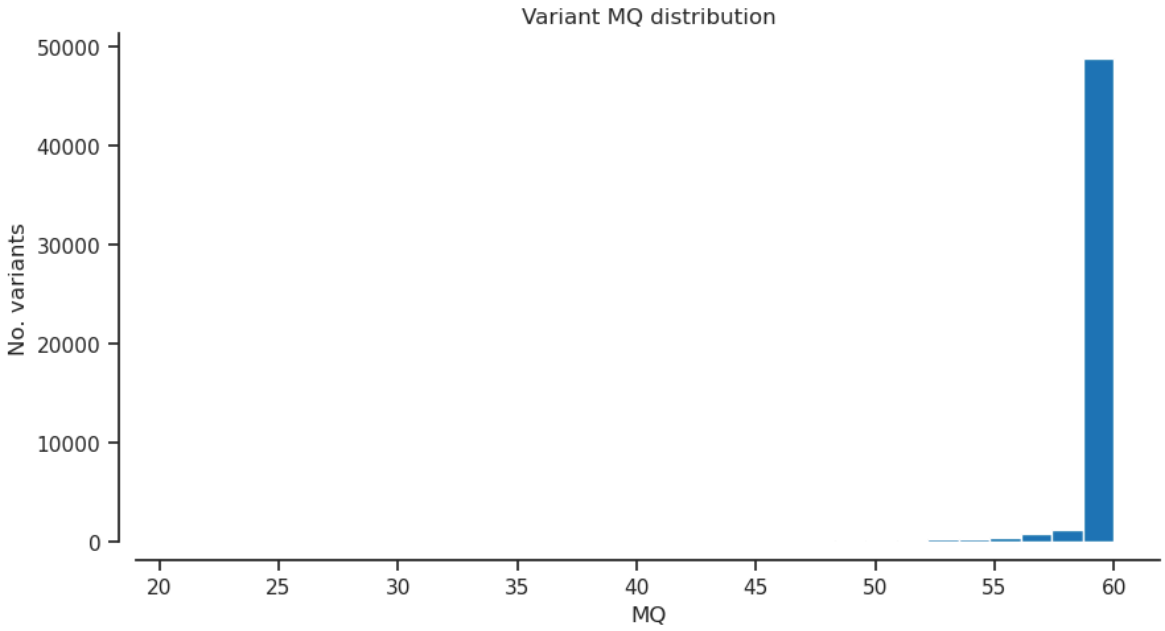
In [13]:
```
filter_expression = '(numalt == 1)'
biallelic_np = rawsnps.query(filter_expression)[:]
biallelic_np
```

Out[13]: <VariantTable shape=(50252,) 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')])>

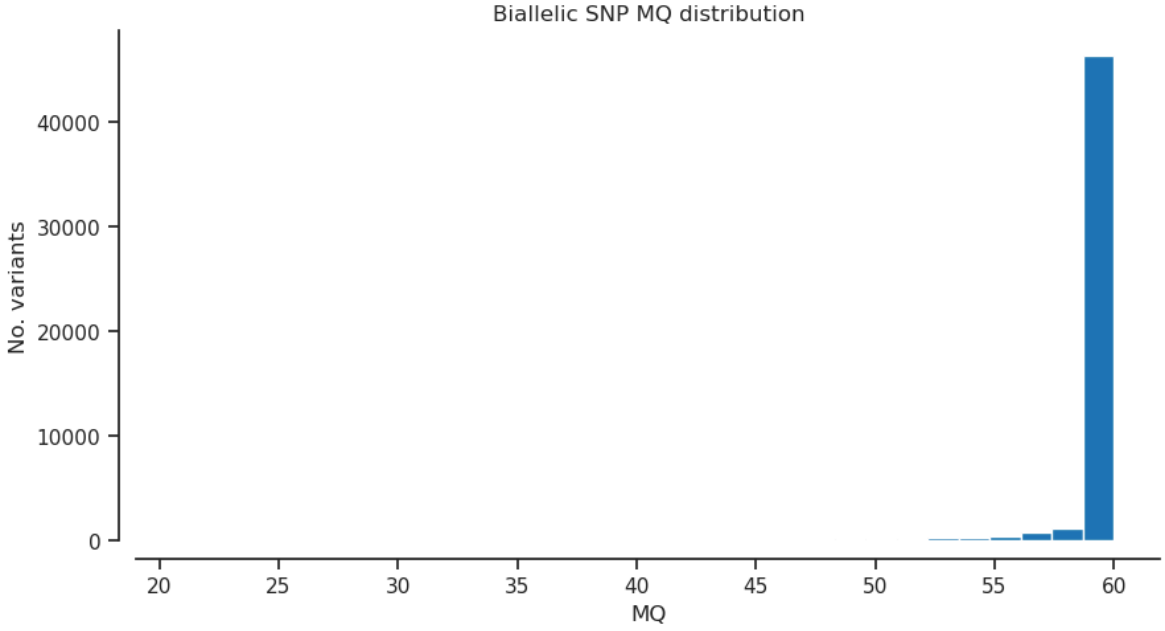
	AC	AF	ALT	AN	BaseQRankSum	CHROM	DP	END	Ex
0	[4 -1 -1]	[0.286 nan nan]	[b'T' b'' b'']	14	1.11	b'aalba5_s00000080'	96	-1	
1	[4 -1 -1]	[0.286 nan nan]	[b'T' b'' b'']	14	0.0	b'aalba5_s00000080'	36	-1	
2	[4 -1 -1]	[0.286 nan nan]	[b'G' b'' b'']	14	0.0	b'aalba5_s00000080'	24	-1	
...									
50249	[6 -1 -1]	[0.1 nan nan]	[b'C' b'' b'']	60	0.0	b'aalba5_s01418300'	156	-1	
50250	[1 -1 -1]	[0.017 nan nan]	[b'A' b'' b'']	60	0.0	b'aalba5_s01418300'	117	-1	
50251	[1 -1 -1]	[0.017 nan nan]	[b'C' b'' b'']	60	0.765	b'aalba5_s01418300'	80	-1	

MQ - RMS mapping quality

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

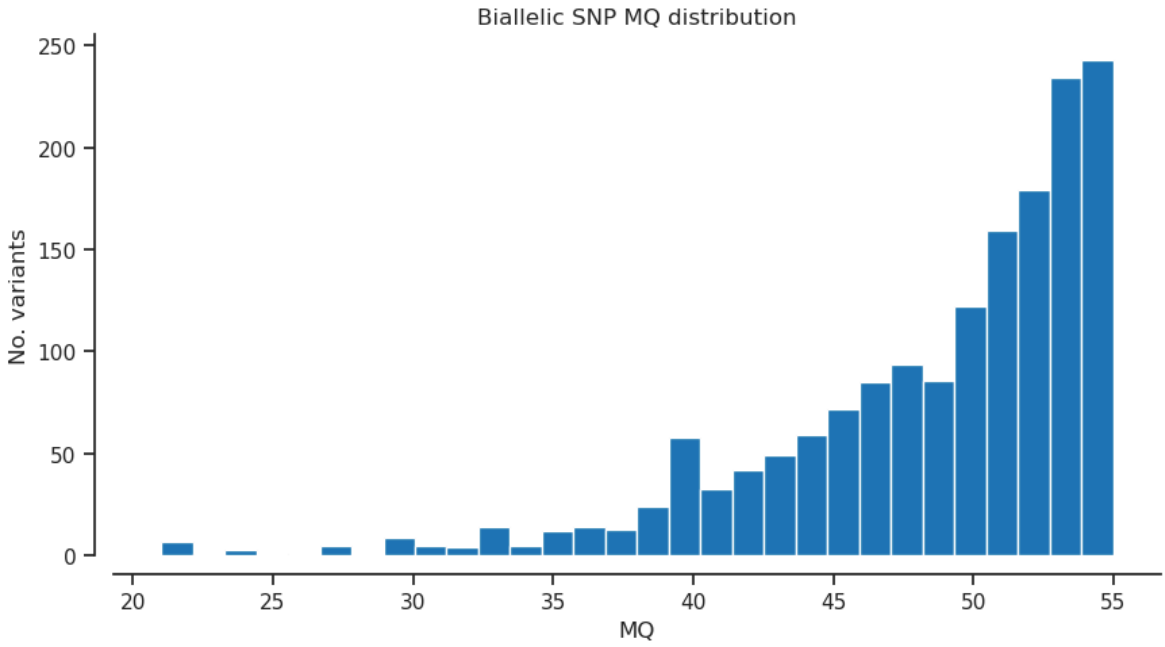


In [15]:
```
plot_hist('MQ','biallelic') # RMS mapping quality
```



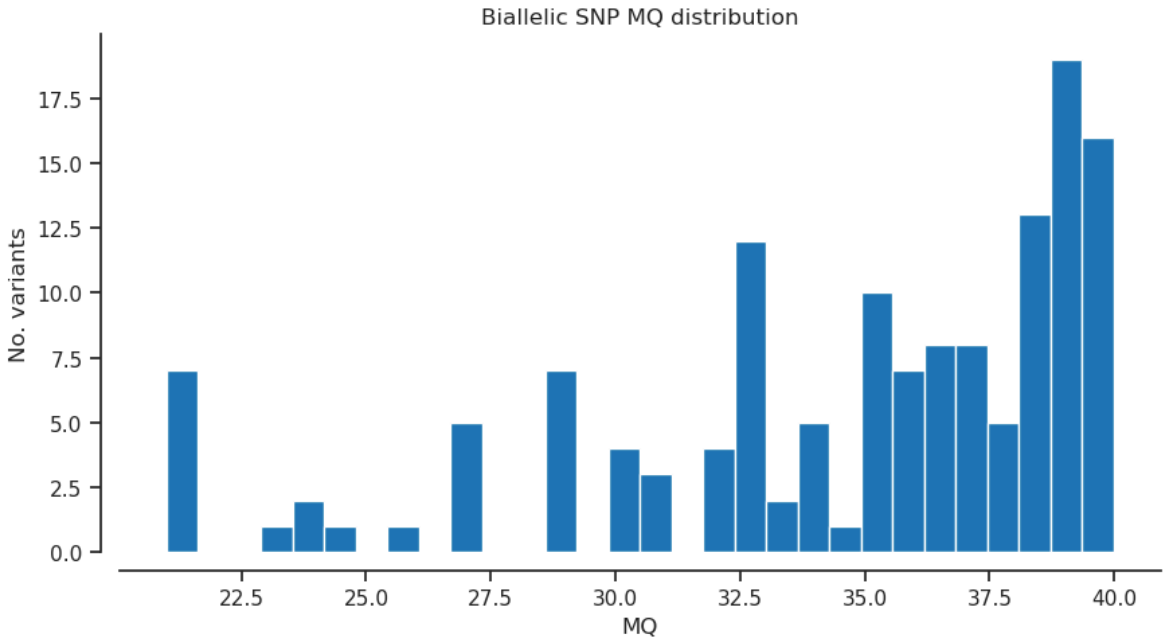
In [16]:
```
filter_expression = '(MQ < 55)'
bi_selection = biallelic_np.query(filter_expression)[:]
#np.count_nonzero(var_selection)
```

In [17]:
```
plot_hist('MQ')
```

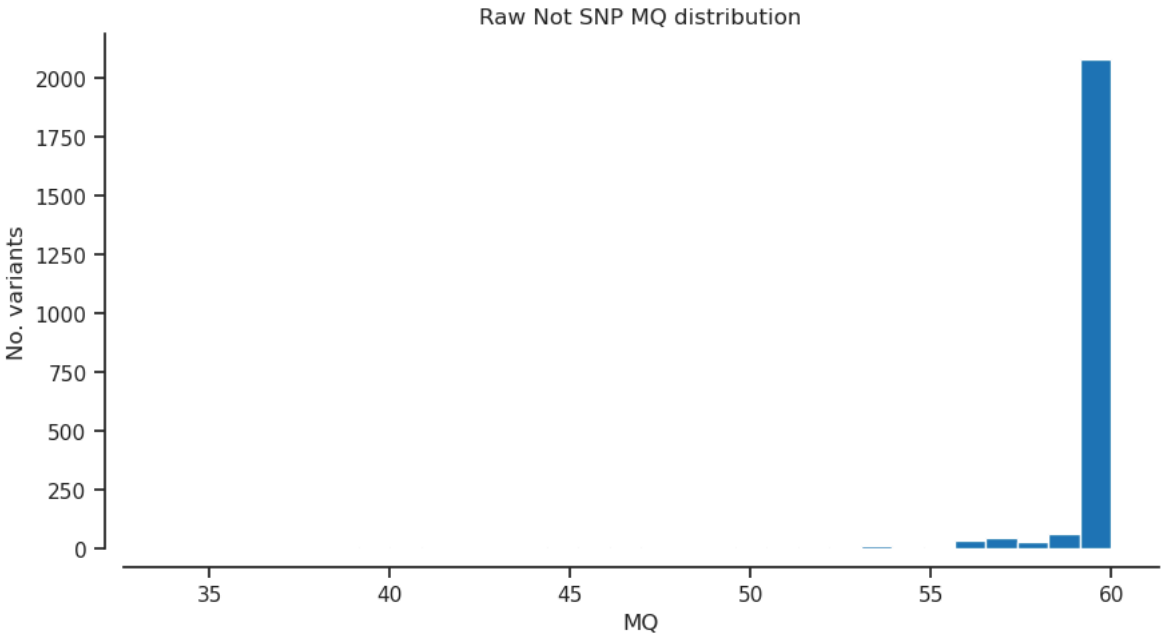


In [18]:
```
filter_expression = '(MQ < 40)'
bi_selection = biallelic_np.query(filter_expression)[:]
```

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

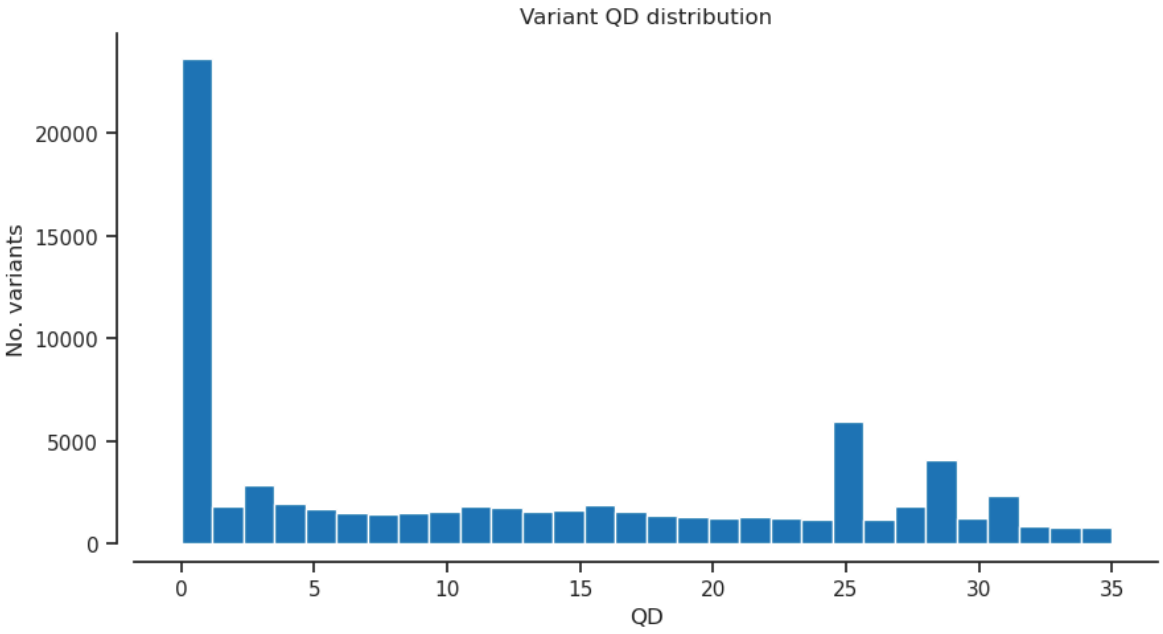


In [20]:
```
plot_hist('MQ','notsnp')
```

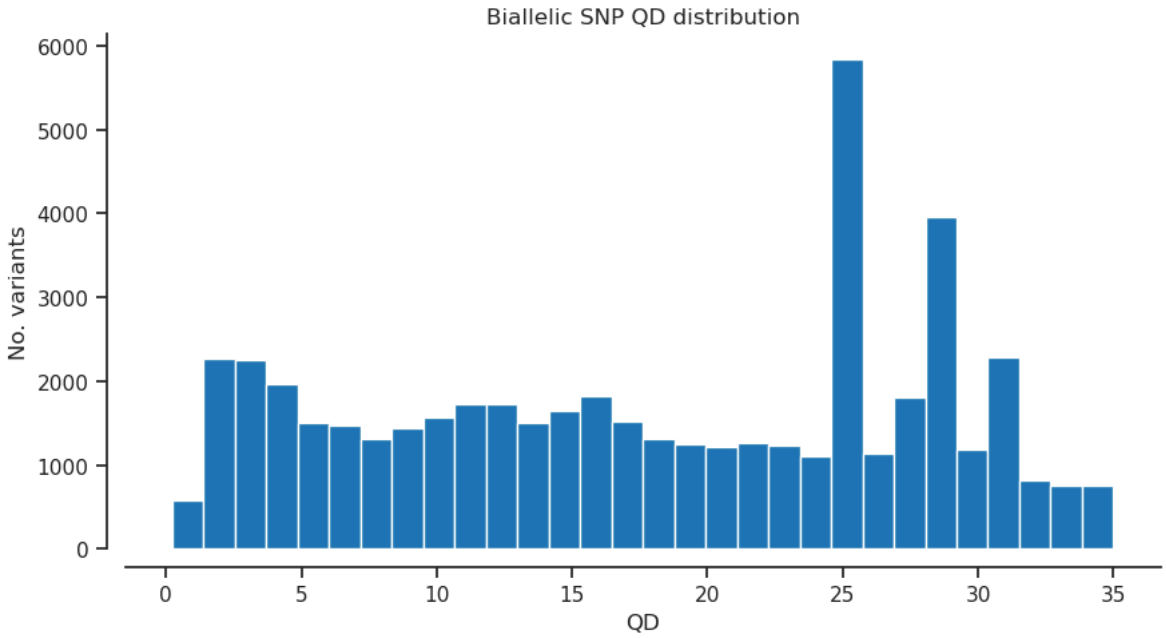


QD - Variant Confidence/Quality by Depth

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

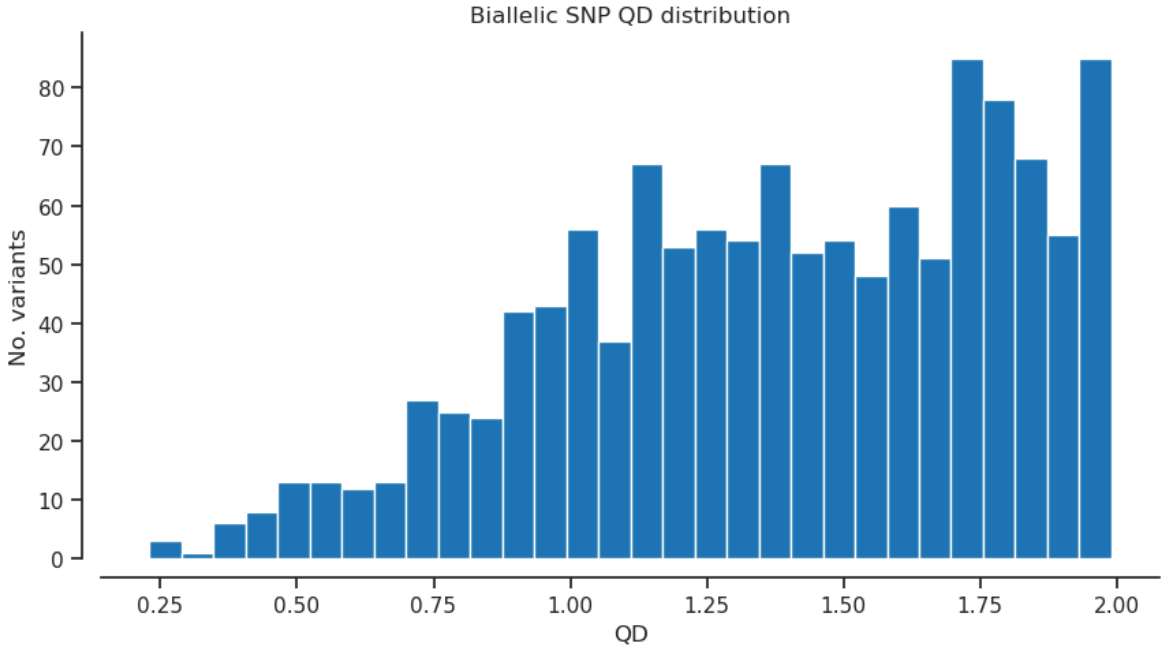


In [22]:
```
plot_hist('QD','biallelic') # Variant Confidence/Quality by Depth
```

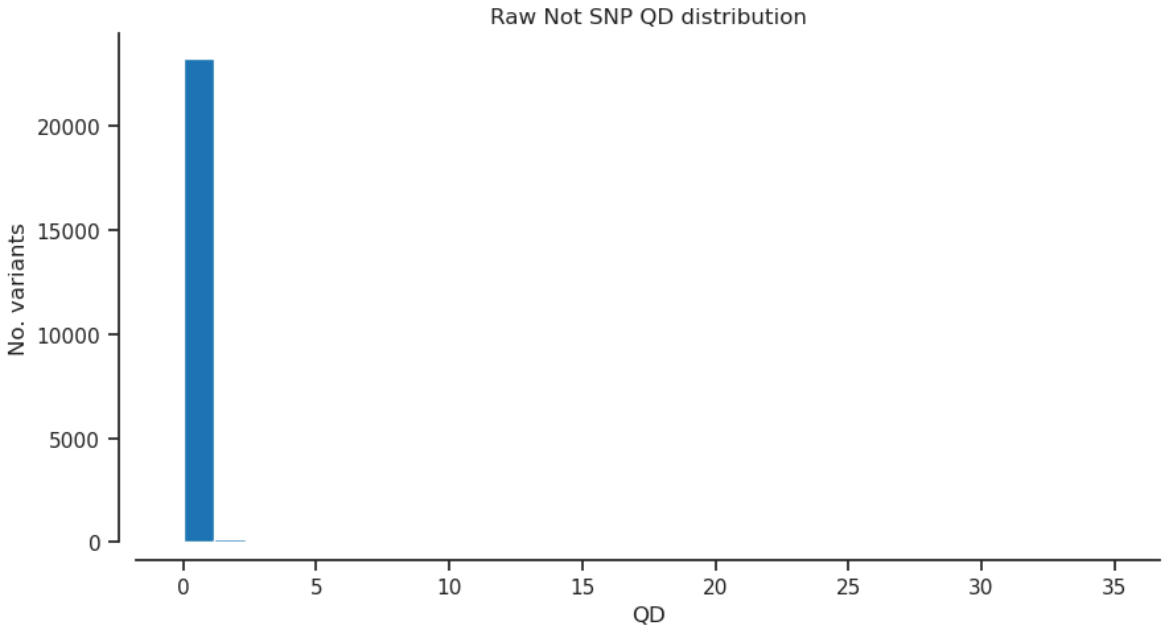


In [23]:
```
filter_expression = '(QD < 2)'
bi_selection = biallelic_np.query(filter_expression)[:]
```

In [24]:
```
plot_hist('QD')
```

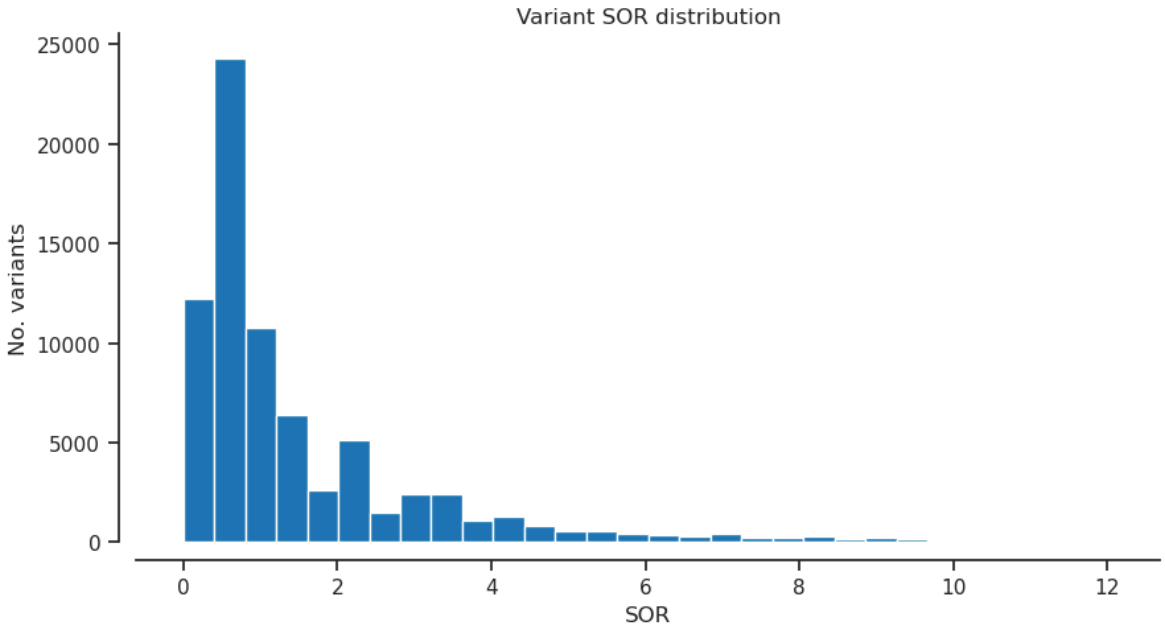


In [25]:
```
plot_hist('QD','notsnp') # Variant Confidence/Quality by Depth
```

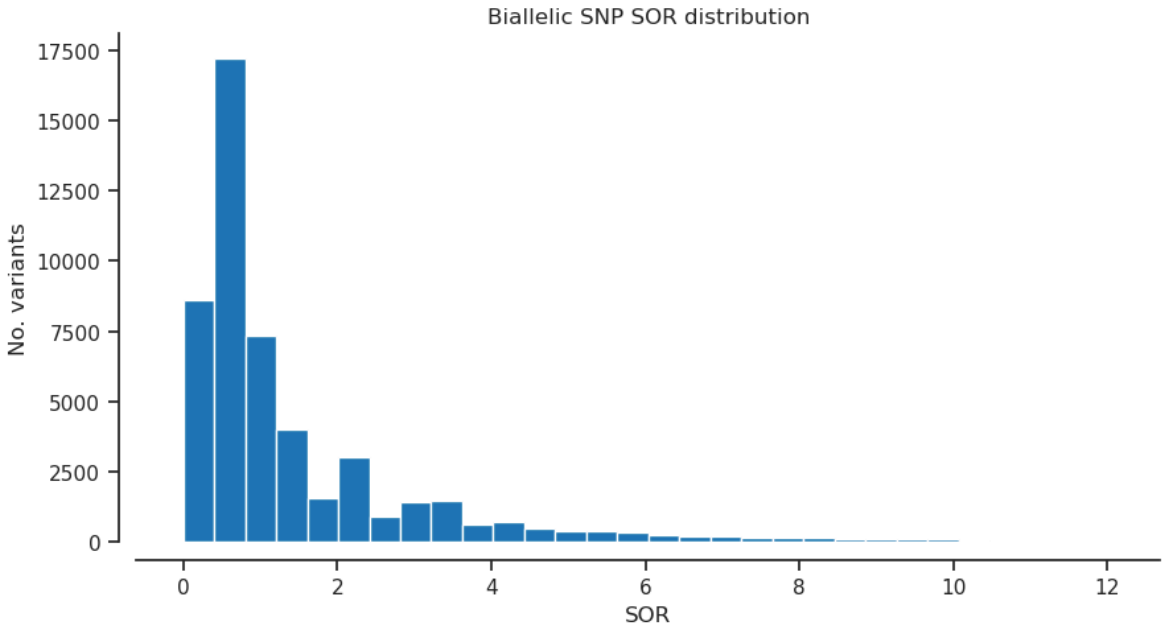


SOR - Symmetric Odds Ratio of 2x2 contingency table to detect strand bias

In [26]:
```
plot_hist('SOR', 'var') # Symmetric Odds Ratio of 2x2 contingency table t
```

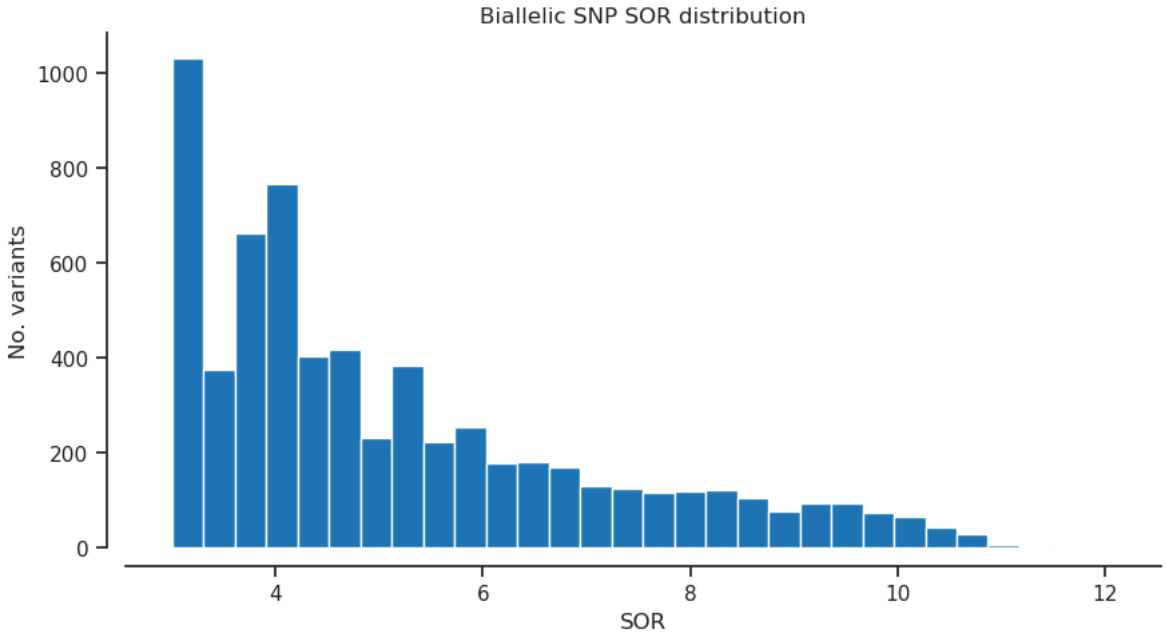


In [27]:
```
plot_hist('SOR','biallelic') # Symmetric Odds Ratio of 2x2 contingency ta
```

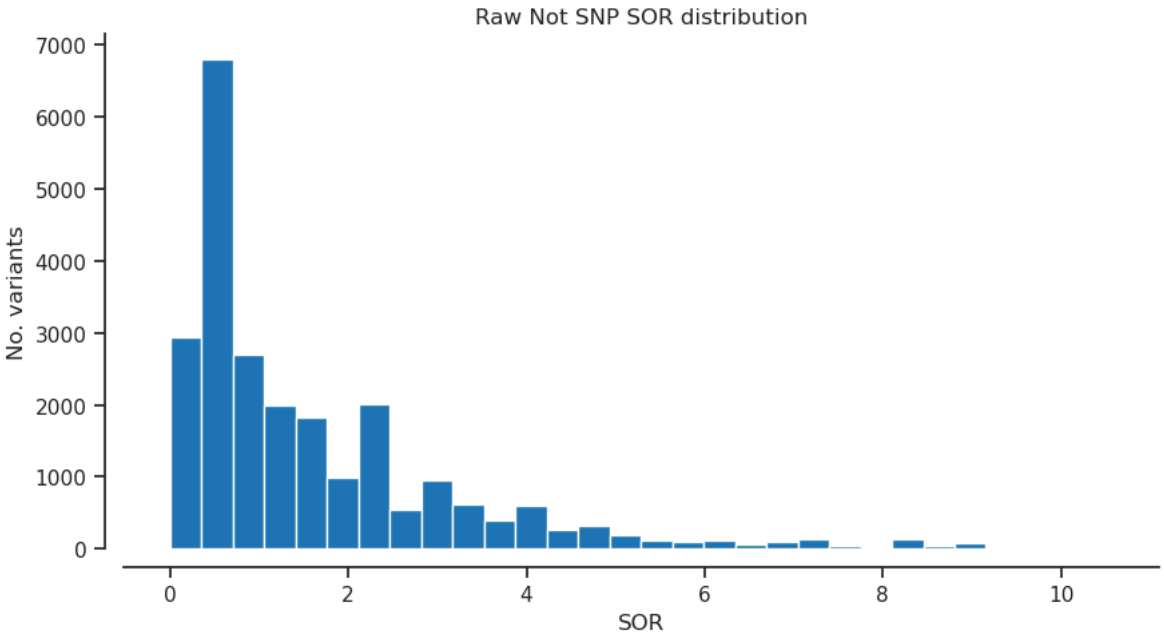


In [28]:
```
filter_expression = '(SOR > 3)'
bi_selection = biallelic_np.query(filter_expression)[:]
```

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

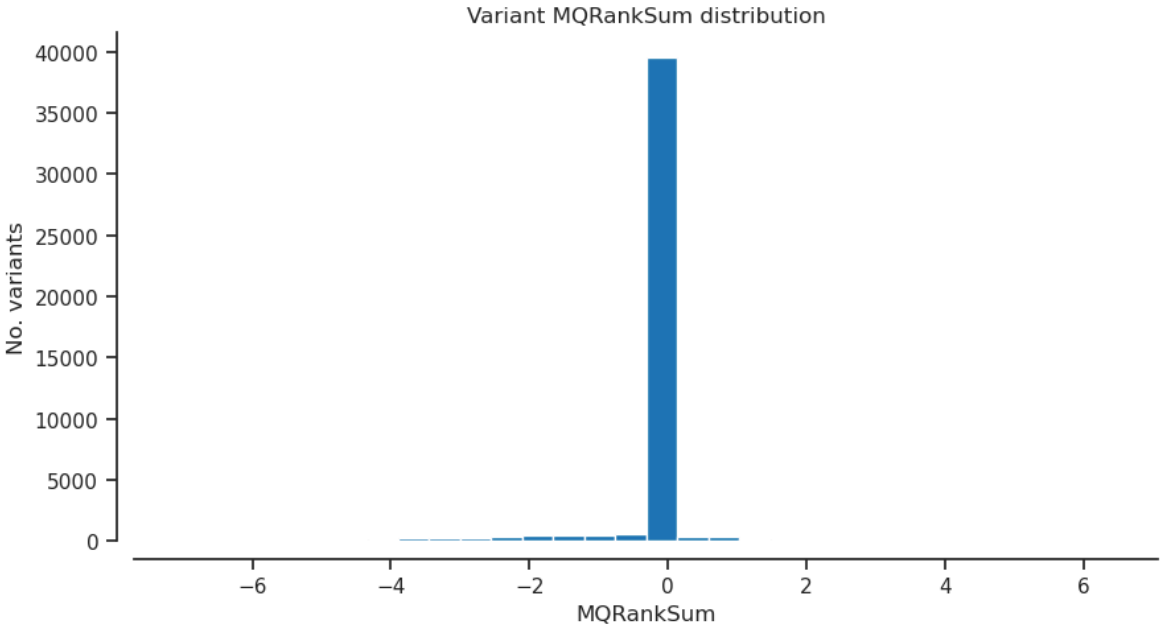


In [30]:
```
plot_hist('SOR','notsnp') # Symmetric Odds Ratio of 2x2 contingency table
```

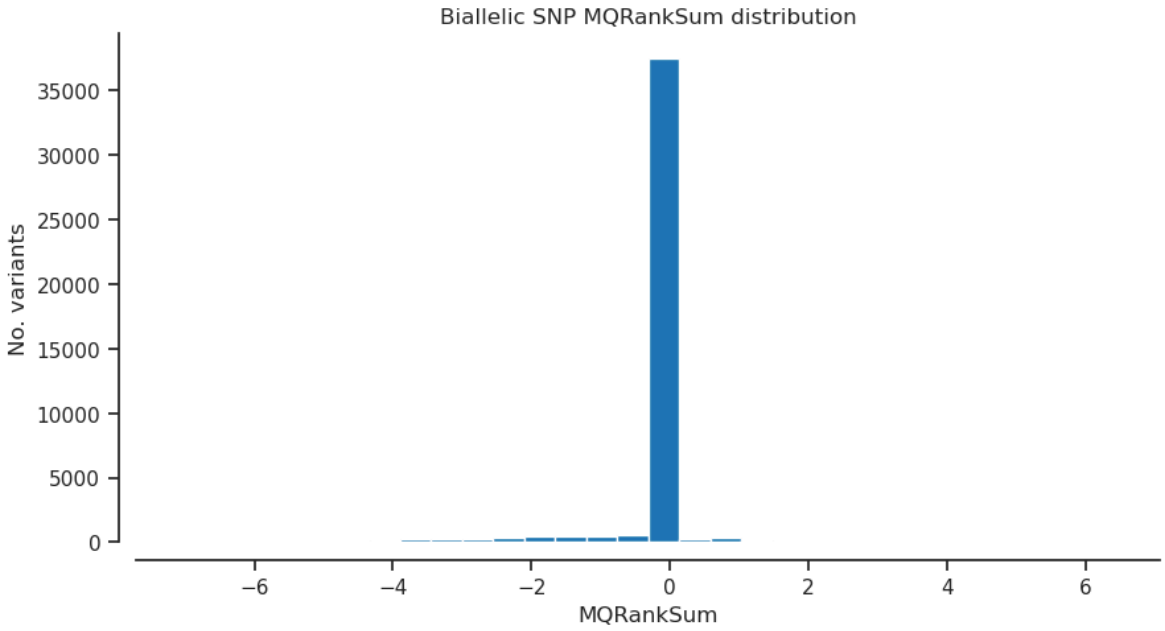


MQRankSum - Z-score From Wilcoxon rank sum test of Alt vs. Ref read mapping qualities

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

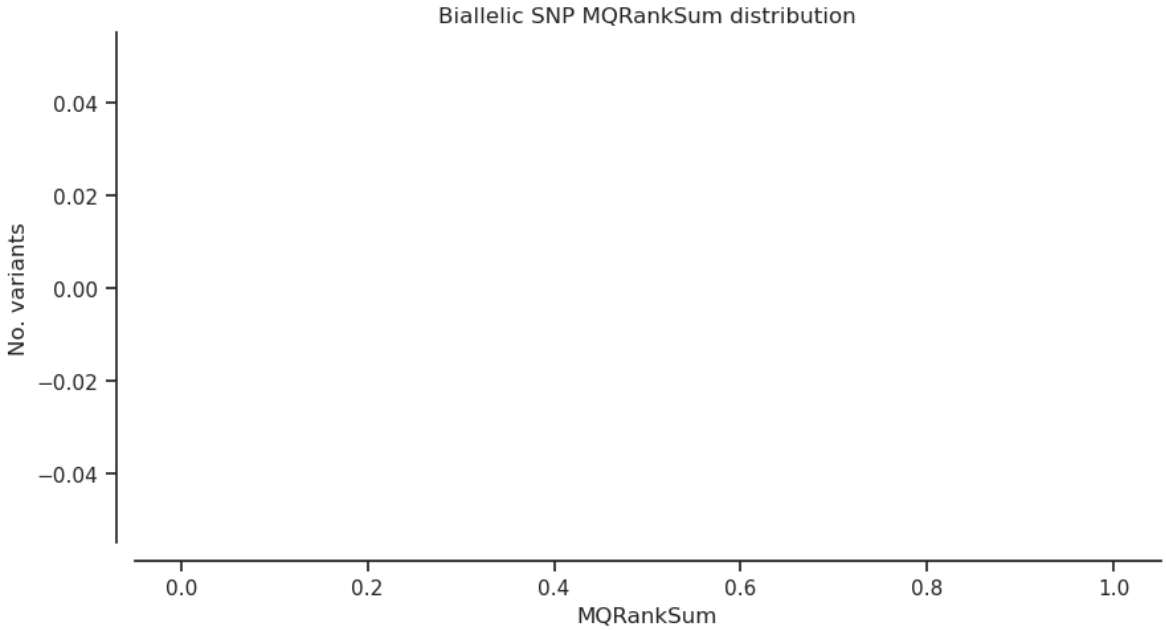


In [32]:
```
plot_hist('MQRankSum','biallelic') # Z-score From Wilcoxon rank sum test
```

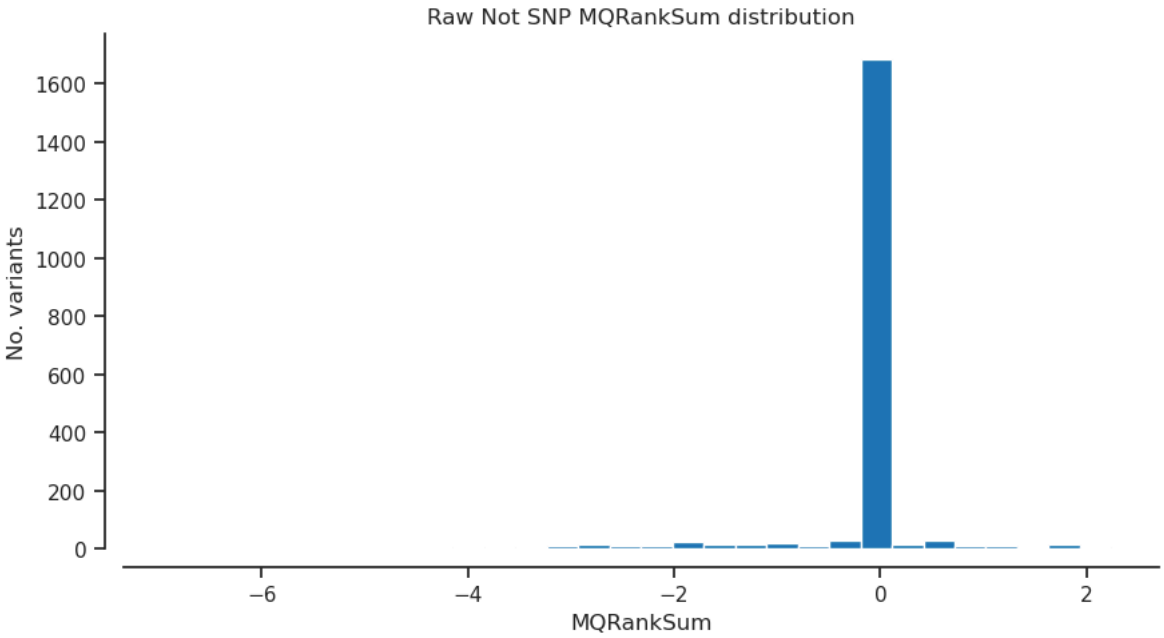


In [33]:
```
filter_expression = '(MQRankSum < -12.5)'
bi_selection = biallelic_np.query(filter_expression)[:]
```

In [34]:
```
plot_hist('MQRankSum') # Z-score From Wilcoxon rank sum test of Alt vs. R
```

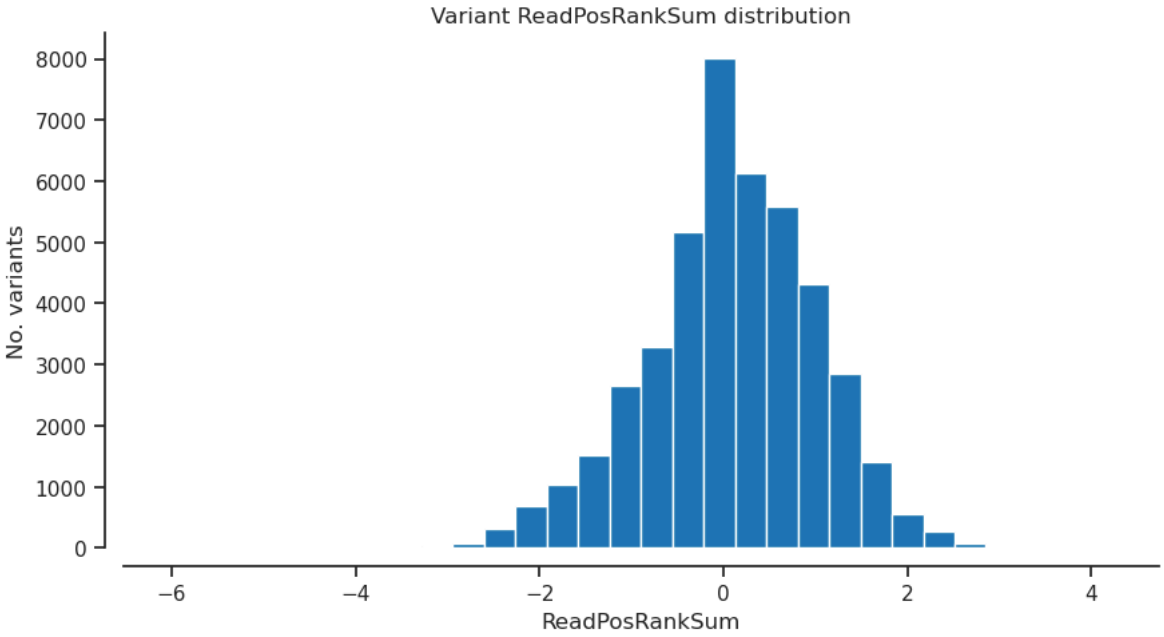


In [35]:
```
plot_hist('MQRankSum','notsnp') # Z-score From Wilcoxon rank sum test of
```

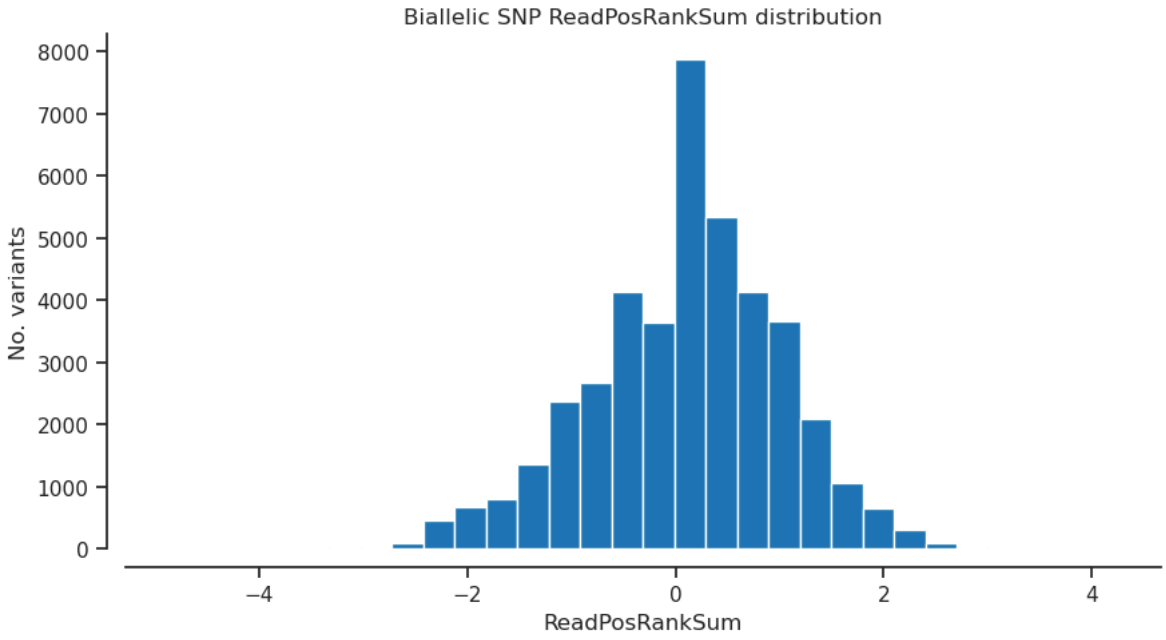


ReadPosRankSum - Z-score from Wilcoxon rank sum test of Alt vs. Ref read position bias

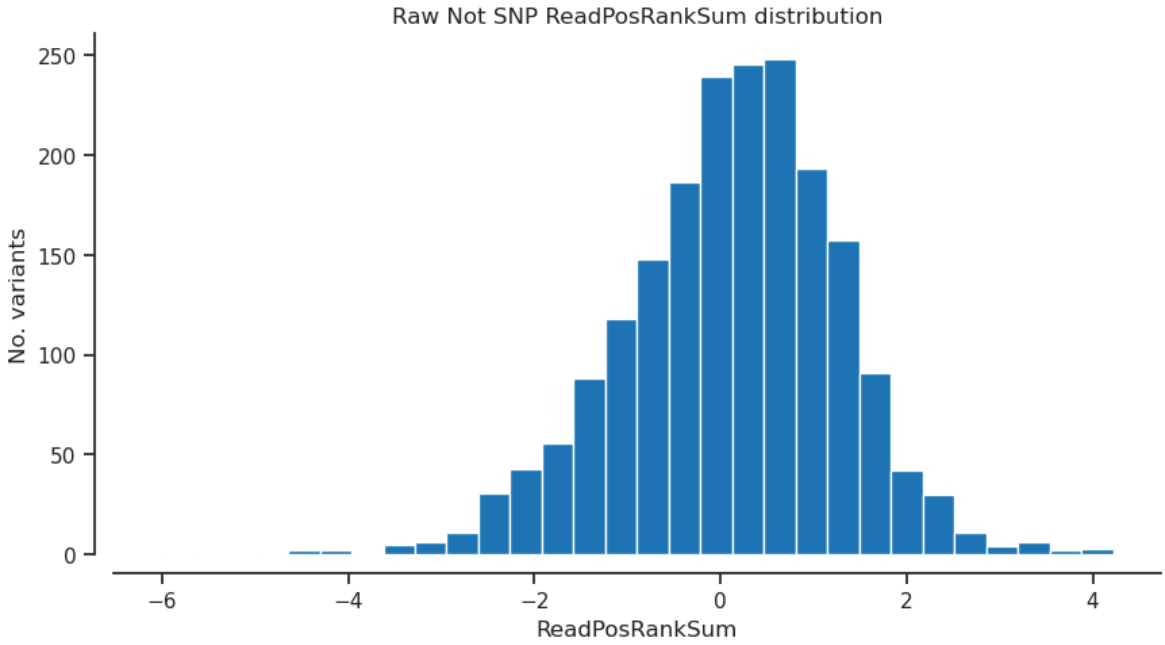
In [36]:
```
plot_hist('ReadPosRankSum','var') # Z-score from Wilcoxon rank sum test o
```



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

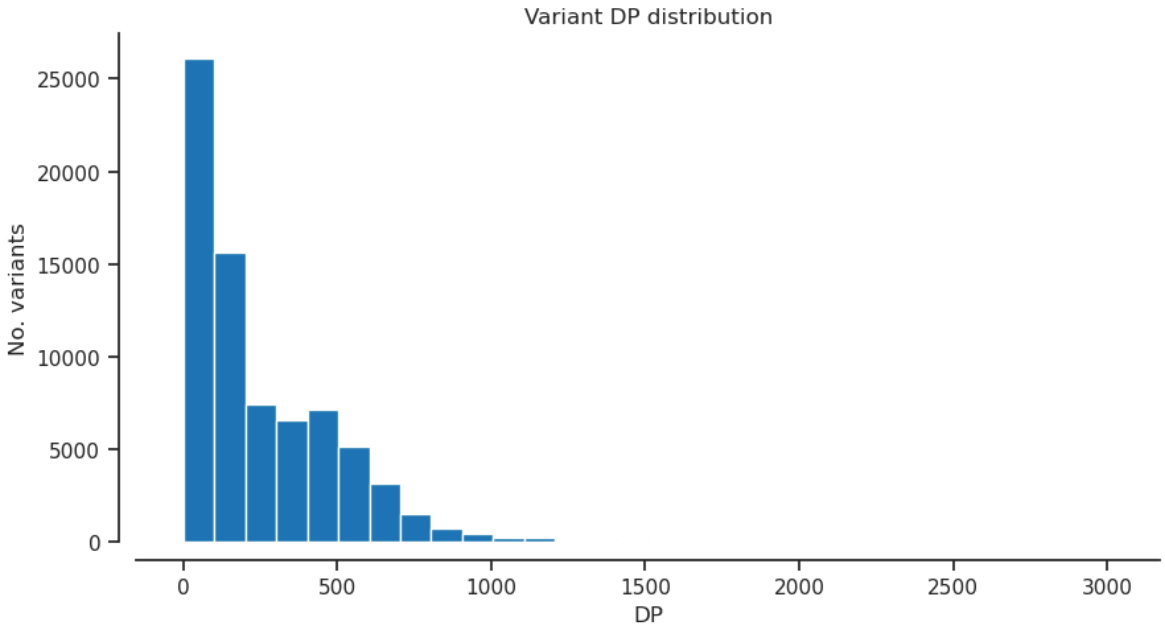


In [38]:
```
plot_hist('ReadPosRankSum','notsnp') # Z-score from Wilcoxon rank sum tes
```

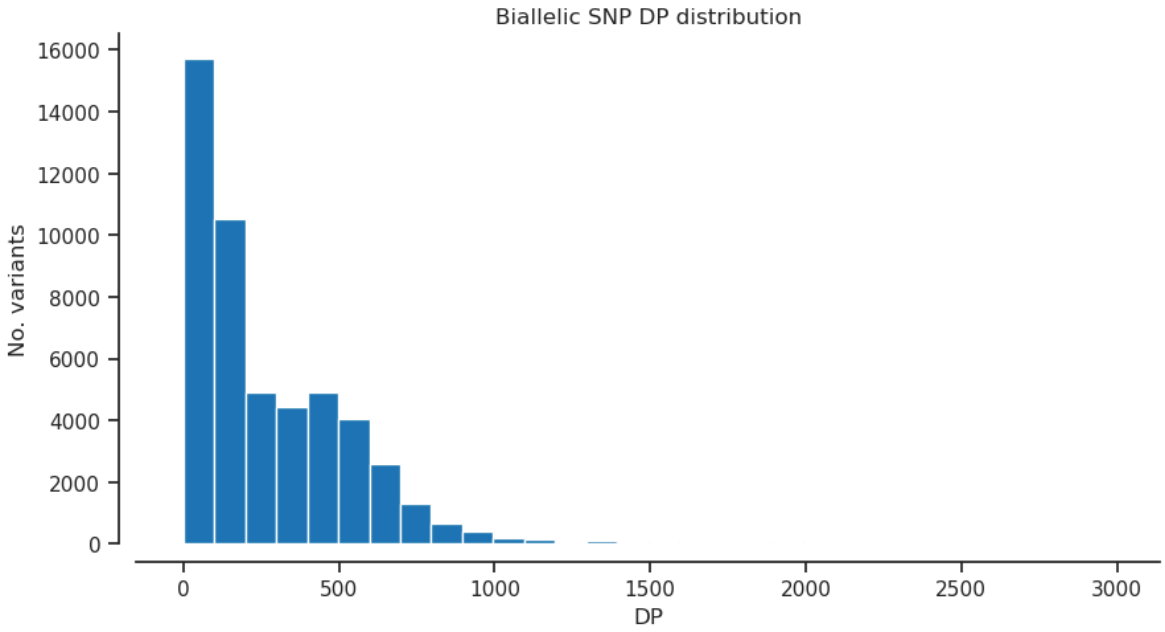


DP - Approximate read depth

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

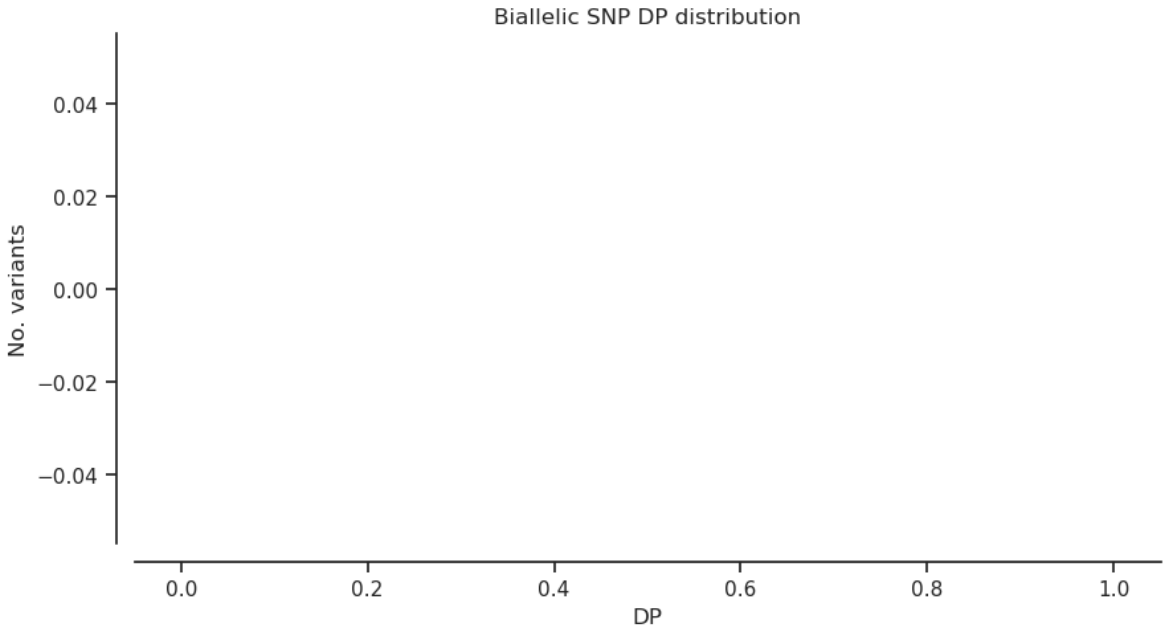


In [40]:
```
plot_hist('DP','biallelic')
```

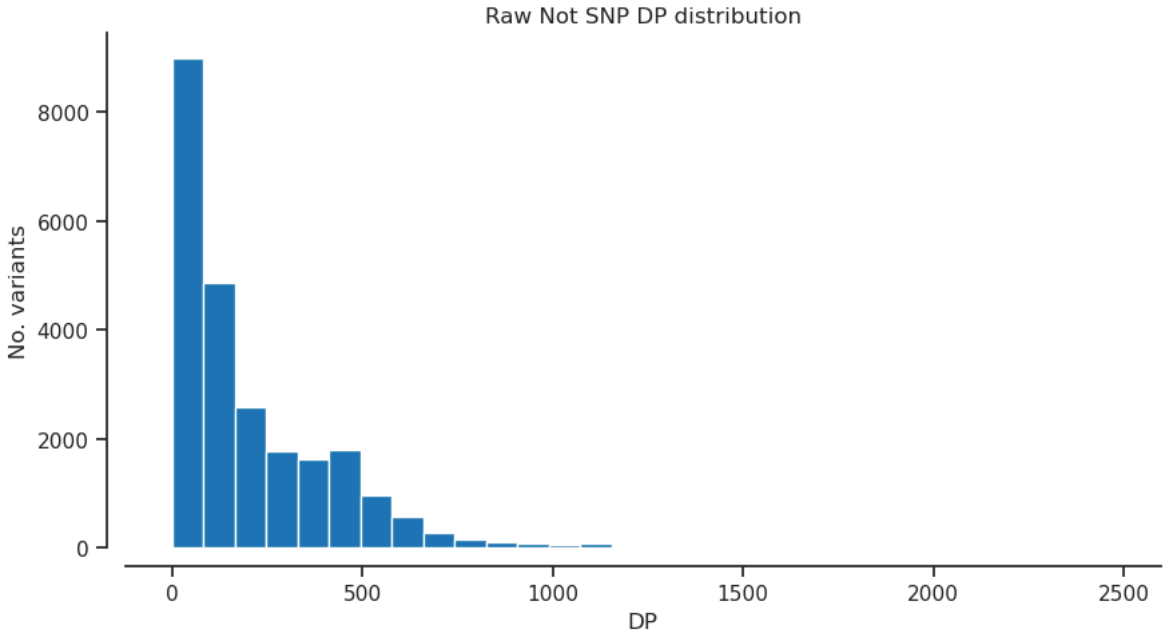


In [41]:
```
filter_expression = '(DP > 20000) & (DP < 40000)'
bi_selection = biallelic_np.query(filter_expression)[:]
```

In [42]:
```
plot_hist('DP')
```

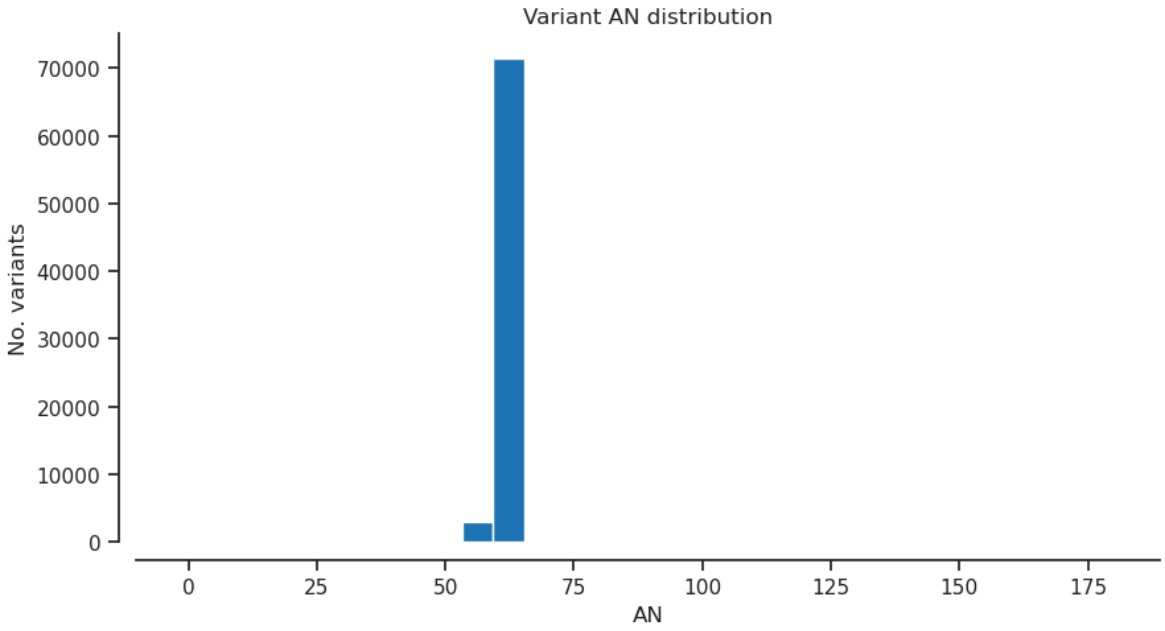


In [43]:
```
plot_hist('DP','notsnp')
```

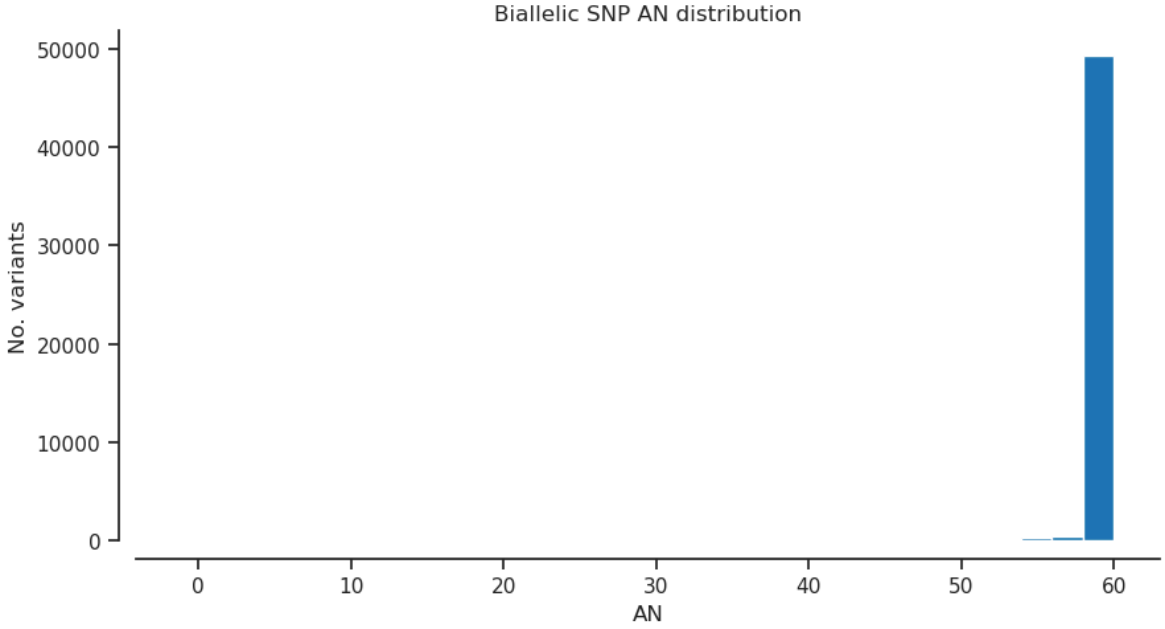


AN - Total number of alleles in called genotypes

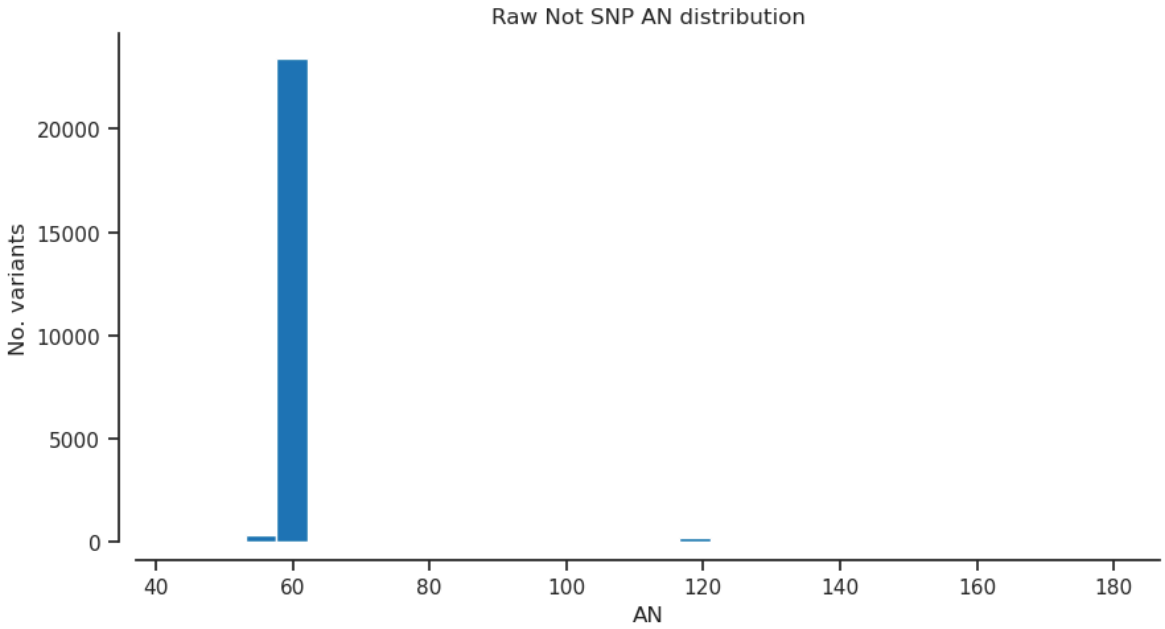
In [44]:
```
plot_hist('AN','var') # Total number of alleles in called genotypes
```



In [45]:
```
plot_hist('AN','biallelic') # Total number of alleles in called genotypes
```



In [46]:
```
plot_hist('AN','notsnp') # Total number of alleles in called genotypes
```



Selected filter

In [47]:
```
# 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[47]:
```
40767
```

Genotype

In [48]:
```
calldata_var = callset_var['calldata']
list(calldata_var)
```

Out[48]:
```
['AD', 'DP', 'GQ', 'GT', 'MIN_DP', 'PGT', 'PID', 'PL', 'PS', 'RGQ', 'S
B']
```

In [49]:
```
genotypes_var = allel.GenotypeChunkedArray(calldata_var['GT'])
genotypes_var
```

Out[49]: <GenotypeChunkedArray shape=(74891, 30, 2) dtype=int8 chunks=(65536, 30, 2) nbytes=4.3M cbytes=456.4K cratio=9.6 compression=gzip compression_opts=1 values=h5py._hl.dataset.Dataset>

	0	1	2	3	4	...	25	26	27	28	29
0	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
1	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
2	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
...						...					
74888	0/0	0/0	0/0	0/0	0/1	...	0/1	0/1	0/0	0/0	0/0
74889	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
74890	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0

In [50]:
```
# using the selected filters set above
gt_filtered_snps = genotypes_var.subset(variant_selection)
gt_filtered_snps
```

Out[50]: <GenotypeChunkedArray shape=(40767, 30, 2) dtype=int8 chunks=(5096, 30, 2) nbytes=2.3M cbytes=581.4K cratio=4.1 compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>

	0	1	2	3	4	...	25	26	27	28	29
0	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
1	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
2	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
...						...					
40764	0/0	0/1	0/0	0/0	0/0	...	0/0	0/0	0/0	0/1	0/1
40765	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
40766	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0

In [51]:
```
# grab the allele counts for the populations
ac = gt_filtered_snps.count_alleles()
ac
```

Out[51]: <AlleleCountsChunkedArray shape=(40767, 4) dtype=int32 chunks=(20384, 4) nbytes=637.0K cbytes=66.8K cratio=9.5 compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>

	0	1	2	3
0	10	4	0	0
1	10	4	0	0
2	10	4	0	0
...		...		
40764	54	6	0	0
40765	59	1	0	0
40766	59	1	0	0

In [52]:
```
ac[:]
```

Out[52]: <AlleleCountsArray shape=(40767, 4) dtype=int32>

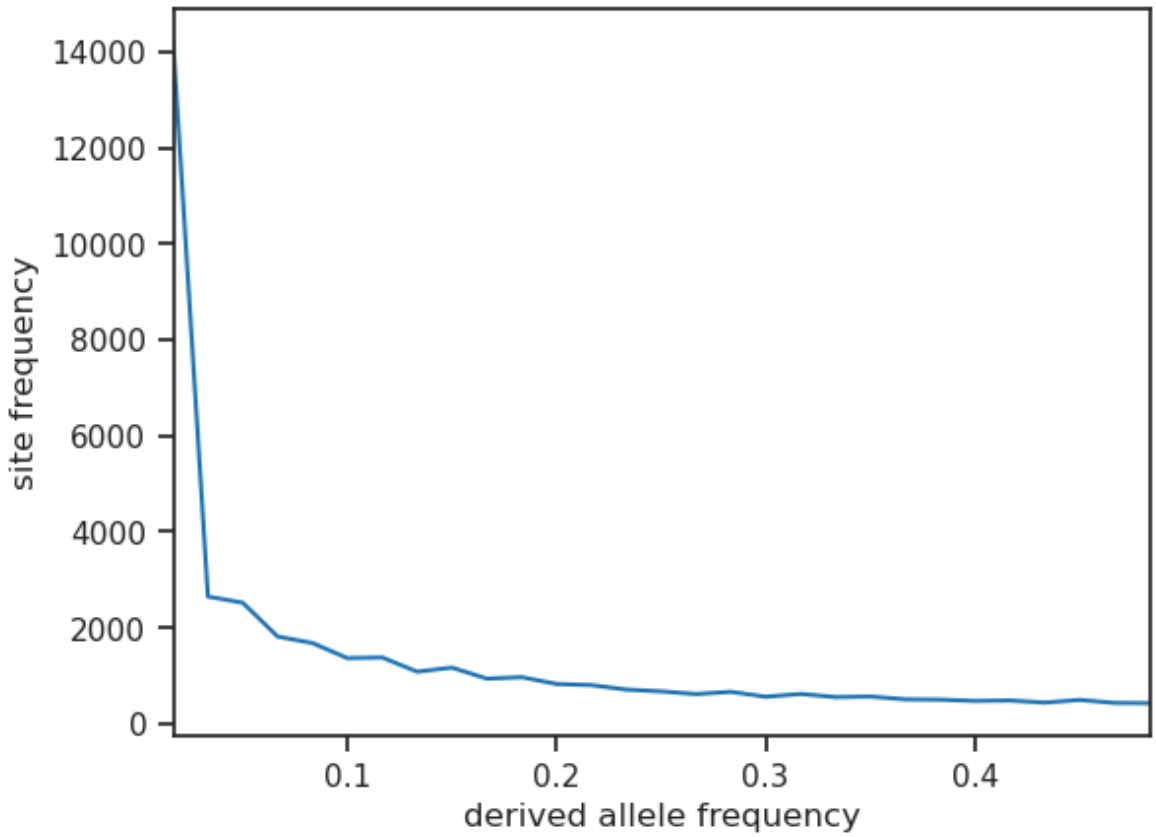
	0	1	2	3
0	10	4	0	0
1	10	4	0	0
2	10	4	0	0
...		...		
40764	54	6	0	0
40765	59	1	0	0
40766	59	1	0	0

In [53]:
```
# 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[53]:
```
array([[10,  4],
       [10,  4],
       [10,  4],
       ...,
       [54,  6],
       [59,  1],
       [59,  1]], dtype=int32)
```

In [54]:
```
# 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[54]:
```
<Axes: xlabel='derived allele frequency', ylabel='site frequency'>
```



In [55]:
```
biallelic = (ac.max_allele() == 1)
###This is the filter expression for biallelic sites
biallelic
```

Out[55]:
```
<ChunkedArrayWrapper shape=(40767,) dtype=bool chunks=(40767,)
   nbytes=39.8K cbytes=2.6K cratio=15.4
   compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shu
ffle': 1, 'blocksize': 0}
   values=zarr.core.Array>
```

In [56]:
```
# select only the biallelic variants
gt_biallelic = gt_filtered_snps.compress(biallelic)
gt_biallelic
```

Out[56]:
<GenotypeChunkedArray shape=(40326, 30, 2) dtype=int8 chunks=(5041, 30, 2) nbytes=2.3M cbytes=570.0K cratio=4.1 compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>

	0	1	2	3	4	...	25	26	27	28	29
0	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
1	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
2	0/0	./.	./.	./.	./.	...	1/1	./.	./.	./.	./.
...						...					
40323	0/0	0/1	0/0	0/0	0/0	...	0/0	0/0	0/0	0/1	0/1
40324	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0
40325	0/0	0/0	0/0	0/0	0/0	...	0/0	0/0	0/0	0/0	0/0

In [57]:
```
n_variants = len(gt_biallelic)
n_variants
```

Out[57]: 40326

In [58]:
```
pc_missing = gt_biallelic.count_missing(axis=0)[:] * 100 / n_variants
pc_het = gt_biallelic.count_het(axis=0)[:] * 100 / n_variants
```

Samples

In [60]:
```
samples_var = callset_var['samples']
samples_var = list(samples_var)
samples_var
```

Out[60]:
```
[b'ITA00252-001',
 b'ITA00252-002',
 b'ITA00252-003',
 b'ITA00252-004',
 b'ITA00252-005',
 b'ITA00252-006',
 b'ITA00252-007',
 b'ITA00252-008',
 b'ITA00252-009',
 b'ITA00252-010',
 b'ITA00252-011',
 b'ITA00252-012',
 b'ITA00252-013',
 b'ITA00252-014',
 b'ITA00252-015',
 b'ITA00252-016',
 b'ITA00252-017',
 b'ITA00252-018',
 b'ITA00252-019',
 b'ITA00252-020',
 b'ITA00252-021',
 b'ITA00252-022',
 b'ITA00252-023',
 b'ITA00252-024',
 b'ITA00252-025',
 b'ITA00252-026',
 b'ITA00252-027',
 b'ITA00252-028',
 b'ITA00252-029',
 b'ITA00252-030']
```

In [63]:
```
samples_fn = '~/scratch/data/Anebrodensis/Abies_nebrodensis_sample_list_s
samples = pandas.read_csv(samples_fn, sep='\t')
samples
```

Out[63]:

	ID	Population
0	ITA00252-001	ITA00252
1	ITA00252-002	ITA00252
2	ITA00252-003	ITA00252
3	ITA00252-004	ITA00252
4	ITA00252-005	ITA00252
5	ITA00252-006	ITA00252
6	ITA00252-007	ITA00252
7	ITA00252-008	ITA00252
8	ITA00252-009	ITA00252
9	ITA00252-010	ITA00252
10	ITA00252-011	ITA00252
11	ITA00252-012	ITA00252
12	ITA00252-013	ITA00252
13	ITA00252-014	ITA00252
14	ITA00252-015	ITA00252
15	ITA00252-016	ITA00252
16	ITA00252-017	ITA00252
17	ITA00252-018	ITA00252
18	ITA00252-019	ITA00252
19	ITA00252-020	ITA00252
20	ITA00252-021	ITA00252
21	ITA00252-022	ITA00252
22	ITA00252-023	ITA00252
23	ITA00252-024	ITA00252
24	ITA00252-025	ITA00252
25	ITA00252-026	ITA00252
26	ITA00252-027	ITA00252
27	ITA00252-028	ITA00252
28	ITA00252-029	ITA00252
29	ITA00252-030	ITA00252

In [64]:
```
samples.Population.value_counts()
```

Out[64]:
```
Population
ITA00252    30
Name: count, dtype: int64
```

In [65]:
```
populations = samples.Population.unique()
populations
###This identifiers come from the metadata file
```

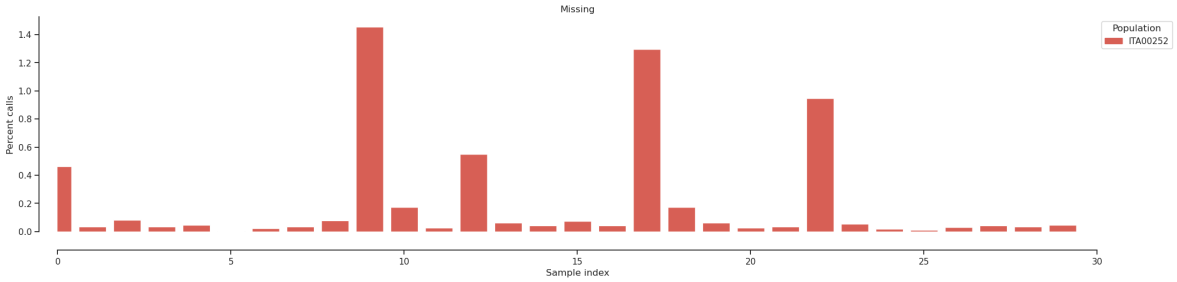
Out[65]:
```
array(['ITA00252'], dtype=object)
```

Gt frequency function

In [66]:
```
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", 1)
    pop2color = {'ITA00252': palette[0]}
    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])]
    ax.legend(handles=handles, labels=['ITA00252'], title='Population',
              bbox_to_anchor=(1, 1), loc='upper left')
```

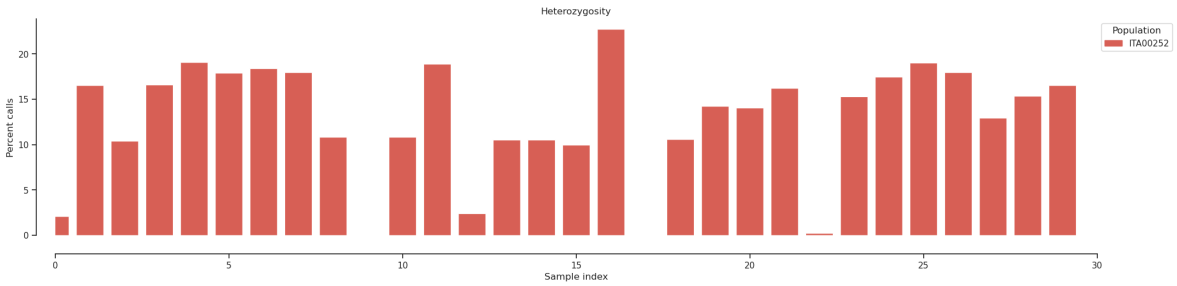
Plot missing

In [67]:
```
plot_genotype_frequency(pc_missing, 'Missing')
```



Plot heterozygosity

In [68]:
```
plot_genotype_frequency(pc_het, 'Heterozygosity')
```



PCA

In [69]:
```
palette = sns.color_palette("hls",1)
pop_colours = {
                'ITA00252': palette[0]
}
```

In [70]:
```
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_colo
                label=pop, markersize=6, mec='k', mew=.5)
    ax.set_xlabel('PC%s (%.1f%%)' % (pc1+1, model.explained_variance_rati
    ax.set_ylabel('PC%s (%.1f%%)' % (pc2+1, model.explained_variance_rati


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 [71]:
```
ac2 = gt_biallelic.count_alleles()
ac2
```

Out[71]:
<AlleleCountsChunkedArray shape=(40326, 2) dtype=int32 chunks=(20163, 2) nbytes=315.0K cbytes=58.5K cratio=5.4 compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shuffle': 1, 'blocksize': 0} values=zarr.core.Array>

	0	1
0	10	4
1	10	4
2	10	4
...	...	
40323	54	6
40324	59	1
40325	59	1

In [72]:
```
flt = (ac2[:, :2].min(axis=1) > 1)
gf = gt_biallelic.compress(flt, axis=0)
gn = gf.to_n_alt()
gn
```

Out[72]:
```
<ChunkedArrayWrapper shape=(26099, 30) dtype=int8 chunks=(6525, 30)
   nbytes=764.6K cbytes=326.0K cratio=2.3
   compression=blosc compression_opts={'cname': 'lz4', 'clevel': 5, 'shu
ffle': 1, 'blocksize': 0}
   values=zarr.core.Array>
```

In [73]:
```
coords1, model1 = allel.pca(gn, n_components=10, scaler='patterson')
```

In [74]:
```
fig_pca(coords1, model1, 'Figure 1. Conventional PCA.')
```

Figure 1. Conventional PCA.