Oct 25, 2022

Introgression

Graham Etherington[1]

[1]The Earlham Institute

dx.doi.org/10.17504/protocols.io.bq7tmznn

Graham Etherington
The Earlham Institute

ABSTRACT

The European polecat (*Mustela putorius*) is a mammalian predator which breeds across much of Europe east to central Asia. In Great Britain, following years of persecution the European polecat has recently undergone a population increase due to legal protection and its range now overlaps that of feral domestic ferrets (*Mustela putorius furo*). During this range expansion, European polecats hybridised with feral domestic ferrets producing viable offspring. Here we carry out population-level whole genome sequencing on domestic ferrets, British European polecats, and European polecats from the European mainland and find high degrees of genome introgression in British polecats outside their previous stronghold, even in those individuals phenotyped as 'pure' polecats.

DOI

dx.doi.org/10.17504/protocols.io.bq7tmznn

EXTERNAL LINK

https://doi.org/10.1093/jhered/esac038

PROTOCOL CITATION

Graham Etherington 2022. Introgression. **protocols.io** https://dx.doi.org/10.17504/protocols.io.bq7tmznn

MANUSCRIPT CITATION please remember to cite the following publication along with this protocol

Etherington GJ, Ciezarek A, Shaw R, Michaux J, Croose E, Haerty W, Palma FD, Extensive genome introgression between domestic ferret and European polecat during population recovery in Great Britain. Journal of Heredity 113(5). doi: 10.1093/jhered/esac038

CREATED

Jan 06, 2021

LAST MODIFIED

Oct 25, 2022

PROTOCOL INTEGER ID

46035

HyDe analyses

1 Note: Due to a bug, HyDe can't handle sequences > 2.1Gb (the size of a 32-bit signed integer), so you'll need to use Sanger snp-sites to create a snp-only dataset.
The following uses the consensus sequences created in Step 2.4 at:
https://www.protocols.io/edit/gatk-nuclear-variant-discovery-and-consensus-assem-bqzgmx3w

1.1 Create an alignment with variant sites only

```
source snp_sites-2.5.1
snp-sites -c -o ferret_polecats_snps.fasta
ferret_polecats.fasta
```

1.2 Hyde requires a tab-delimited sample-population file, e.g.

sample.map

```
domestic_LIB8733        domestic
domestic_LIB8734        domestic
...
euro_LIB18989   euro
euro_LIB18990   euro
euro_LIB18991   euro
...
hybrid_LIB21971 hybrid
hybrid_LIB21972 hybrid
hybrid_LIB21973 hybrid
uk_euro_LIB21974        welsh
uk_euro_LIB21975        welsh
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
uk_euro_LIB22032        welsh
uk_euro_LIB23764        welsh
uk_euro_S07     english
uk_euro_S08     english
...
weasel  out
```

1.3 Next, the input to Hyde is Phylip, so we need to change our .fasta file into a .ph file. The order of the sequences input into Hyde has to be in the same order as the sample.map, so we also need to sort the sequences whilst we're re-formatting them.

sort_fasta_to_phylip.py3

```
from Bio import SeqIO
import os
import sys


#Takes a fasta file and a Hyde sample map and orders the
sequences in the same order as the sample map, and
writes output to phylip
infile = sys.argv[1] #the input fasta file
outfile = sys.argv[2] #the sorted output phylip file
sample_map = sys.argv[3] #the sample map (two columns,
tab delimited, fasta sample name in the first column,
population name (ignored) in the second)

n_seqs=0
seq_length=0

with open(infile, "r") as in_handle:
        for record in SeqIO.parse(in_handle, "fasta"):
                n_seqs+=1
                seq_length=len(record)
in_handle.close()


records = SeqIO.index(infile, "fasta")

fout = open(outfile,"w")
fout.write(" "+str(n_seqs)+"\t"+str(seq_length)+"\n")

with open(sample_map) as f:
    for line in f:
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
        clade=line.split()
        sample=clade[0]
        record=records[sample]
        fout.write(record.id+"\t"+str(record.seq)+"\n")
fout.close()
```

Run with:

```
python3 ~/python_scripts/fasta/sort_fasta_to_phylip.py3
ferret_polecats_snps.fasta ferret_polecats_snps.ph
sample.map
```

1.4 Run HyDe

```
INFILE=ferret_polecats_snps.ph
###get the first line of the phylip file to get the
number of
### taxa and sequence length for the -n and -s
paramters. There's always 7 taxa (-t)
firstline=$(head -n 1 $INFILE)
arr=($firstline)
spscount=${arr[0]}
seqlength=${arr[1]}
#echo Outfile prefix = $OUTFILEPX.hyde
echo Number of species = $spscount
echo Sequence length = $seqlength
source hyde-0.4.3
run_hyde_mp.py --infile $INFILE --map sample.map   --
outgroup out --num_ind $spscount  --num_sites $seqlength
--num_taxa 5  --threads 4 --prefix polecats
```

1.5 Once we have our trios that involve hybridisation/introgression, we can examine every individual in those trios.

```
INFILE=ferret_polecats_snps.ph
OUTFILEPX=$(basename $INFILE)
###get the first line of the phylip file to get the
number of
### taxa and sequence length for the -n and -s
paramters.
firstline=$(head -n 1 $INFILE)
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
arr=($firstline)
spscount=${arr[0]}
seqlength=${arr[1]}
echo Outfile prefix = $OUTFILEPX.hyde
echo Number of speies = $spscount
echo Sequence length = $seqlength
source hyde-0.4.3
individual_hyde_mp.py --infile $INFILE --map
k7_sample.map   --outgroup out --num_ind $spscount  --
num_sites $seqlength --num_taxa 7  --threads 4 --prefix
polecats_individual -tr  polecats.hyde-out-filtered.txt
```

DSuite analyses

2 Dsuite calculates ABBA-BABA tests for quartets of samples, providing the D-statistic
Weasel will be used as the Outgroup
Firstly I need a population file. This is practically identical to the Hyde sample map (see sample.map above), other than having the keyword 'Outgroup' as the population name for weasel.
Dsuite can be run with a tree. The tree is a population-level one. Here's mine.

```
((((((welsh,english),euro),domestic),(bff,steppe)),Outgroup))
```

2.1 First, I want to select only bi-allelic snps

```
source gatk-4.1.3.0
gatk --java-options "-Xmx10g -Xms10g" SelectVariants -R
MusPutFur1.0_bionano.fasta -V
all_samples_genotyped_snps.vcf -O
all_samples_genotyped_biallelic_snps.vcf  --restrict-
alleles-to BIALLELIC -select-type SNP
```

2.2 Run Dsuite

```
source dsuite-0.4_r28
source gcc-6.2.0
Dsuite Dtrios -t tree.nwk
all_samples_genotyped_biallelic_snps.vcf sample.map -o
ferret_polecat_tree
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

Allele sharing

3 I want to extract loci where all the domestic ferret samples are homozygous ref and all the high-coverage (>10x coverage) European polecat samples are homozygous var (alt), using the GenotypeGVCFs file that I generated from the all_samples_GenomicsDB. I also want all the sites to have at least 5x coverage.

select_dom_hom_polcat_alt_hiqual.sh

```
source jre-8u92
source gatk-4.1.3.0

gatk --java-options "-Xmx150g -Xms150g -XX:ParallelGCThreads=2"
SelectVariants -R MusPutFur1.0_bionano.fasta -sn domestic_LIB8733 -
sn  domestic_LIB8734 -sn domestic_LIB8735 -sn domestic_LIB8736 -sn
domestic_LIB8737 -sn domestic_LIB8738 -sn domestic_LIB8739 -sn
domestic_LIB8740 -sn euro_LIB18989 -sn euro_LIB18990 -sn
euro_LIB18991  -sn euro_LIB18992 -sn euro_LIB18993 -sn
euro_LIB18994 -sn euro_LIB18995  -select 'AF > 0.25 &&
vc.getGenotype("domestic_LIB8733").isHomRef() &&
vc.getGenotype("domestic_LIB8734").isHomRef() &&
vc.getGenotype("domestic_LIB8735").isHomRef() &&
vc.getGenotype("domestic_LIB8736").isHomRef() &&
vc.getGenotype("domestic_LIB8737").isHomRef() &&
vc.getGenotype("domestic_LIB8738").isHomRef() &&
vc.getGenotype("domestic_LIB8739").isHomRef() &&
vc.getGenotype("domestic_LIB8740").isHomRef() &&
vc.getGenotype("euro_LIB18989").isHomVar() &&
vc.getGenotype("euro_LIB18990").isHomVar() &&
vc.getGenotype("euro_LIB18991").isHomVar() &&
vc.getGenotype("euro_LIB18992").isHomVar() &&
vc.getGenotype("euro_LIB18993").isHomVar() &&
vc.getGenotype("euro_LIB18994").isHomVar() &&
vc.getGenotype("euro_LIB18995").isHomVar() &&
vc.getGenotype("domestic_LIB8733").getDP() > 5 &&
vc.getGenotype("domestic_LIB8734").getDP() > 5 &&
vc.getGenotype("domestic_LIB8735").getDP() > 5 &&
vc.getGenotype("domestic_LIB8736").getDP() > 5 &&
vc.getGenotype("domestic_LIB8737").getDP() > 5 &&
vc.getGenotype("domestic_LIB8738").getDP() > 5 &&
vc.getGenotype("domestic_LIB8739").getDP() > 5 &&
vc.getGenotype("domestic_LIB8740").getDP() > 5 &&
vc.getGenotype("euro_LIB18989").getDP() > 5 &&
vc.getGenotype("euro_LIB18990").getDP() > 5 &&
vc.getGenotype("euro_LIB18991").getDP() > 5 &&
vc.getGenotype("euro_LIB18992").getDP() > 5 &&
vc.getGenotype("euro_LIB18993").getDP() > 5 &&
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
vc.getGenotype("euro_LIB18994").getDP() > 5 &&
vc.getGenotype("euro_LIB18995").getDP() > 5' --restrict-alleles-to
BIALLELIC -V all_samples_genotyped_snps.vcf -O
dom_ep_fixed_hiqual.vcf
```

3.1 A quick check to make sure that all the sites are high quality.

```
#get the QUAL and NSAMPLES fields from each call
gatk   VariantsToTable -V dom_ep_fixed.vcf -F CHROM -F
POS -F TYPE -F QUAL -F  NSAMPLES -F NCALLED -F MULTI-
ALLELIC -O dom_ep_fixed.table
#calculate the average per-sample QUAL and print
anything less than 30
awk 'FNR==1{next}   {if ($4/$5 < 30) print $0}'
dom_ep_fixed.table
```

Count the number of loci

```
grep -c "^[^#;]" dom_ep_fixed.vcf
```

3.2 Fixed alleles in other samples

Using the dom_ep_fixed.vcf file as an interval file, we can now make sample-specific vcf files that only contain the 8288 sites that are fixed.
In the script bdlow $sample is the sample_name found in the vcf file, $vcf is the output vcf
e.g. sbatch make_vcfs.sh domestic_LIB8733 ./m_putorius/furo/LIB8733.vcf.gz

```
sample=$1 #the sample_name found in
all_samples_genotyped_snps.vcf
vcf=$2 #the output VCF file

source jre-8u92
source gatk-4.1.3.0_spark
srun  gatk --java-options "-Xmx5g -Xms5g -
XX:ParallelGCThreads=2"  SelectVariants -R
../../reference/MusPutFur1.0_bionano.fasta -sn $sample
-V all_samples_genotyped_snps.vcf -L dom_ep_fixed.vcf -O
$vcf
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

Next, compare the genotypes and alleles in the output vcf files to those in dom_ep_fixed.sh using a custom python script, where the input is the same as above.

```
import vcf
import sys
import imp
import collections



vcf_master = sys.argv[1]#the vcf file to compare the
other vcf file to
vcf_compare = sys.argv[2]#the vcf file to compare
sample_name = sys.argv[3] #the sample to allocate

#open the infile
master = vcf.VCFReader(filename=vcf_master)
compare = vcf.VCFReader(filename=vcf_compare)


#create a dictionary which will use tuples as keys, e.g.
vcf_master_dict[chr01][999] = GT
vcf_master_dict = {}
n_master=0
n_compare=0
n_matching=0
n_matching_het=0
n_non_matching_het=0
n_hom_ref=0
n_hom_alt=0
n_not_called=0
n_not_accouted=0
n_not_in_compare=0

#loop thru each record in the master vcf
for record in master:
    alleles=record.alleles
    #print(alleles)
    vcf_master_dict[(record.CHROM, record.POS)] =
alleles
    n_master+=1

#loop thru each record in the compare vcf
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
for record in compare:
    n_compare+=1
    if (record.CHROM, record.POS) in vcf_master_dict:
        n_matching+=1
        for sample in record.samples:
            #check to confirm that the sample name
(sys.argv[3]) is present in our compare vcf
            if sample.sample == sample_name:
                current_genotype = sample["GT"]
                #if it's heterozygous
                if current_genotype == '0/1' or
current_genotype == '0|1':
                    if (vcf_master_dict[(record.CHROM,
record.POS)] == record.alleles):
                        n_matching_het+=1
                    else:
                        n_non_matching_het+=1

                #if it's homozygous ref
                elif current_genotype == '0/0' or
current_genotype == '0|0':
                    alleles =
vcf_master_dict[(record.CHROM, record.POS)]
                    ref = alleles[0]
                    if (ref == record.REF):
                        n_hom_ref+=1

                #if it's homozygous alt
                elif current_genotype == '1/1' or
current_genotype == '1|1':
                    if (vcf_master_dict[(record.CHROM,
record.POS)] == record.alleles):
                        n_hom_alt+=1
                #if it's not called
                elif current_genotype == './.' or
current_genotype == '.|.':
                    n_not_called+=1

                else:
                    n_not_accouted+=1
                    sys.stderr.write("\nThe following
genotype was not found in the master vcf: ")
                    sys.stderr.write(str(record))

sys.stderr.write(str(current_genotype))

sys.stderr.write(str(vcf_master_dict[record.CHROM,
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
record.POS]))

    else:
        n_not_in_compare+=1
        sys.stderr.write("\nThe following record was not
found in the master vcf: ")
        sys.stderr.write(str(record))


#n_combined = n_matching_het + n_hom_ref + n_hom_alt
part_het = float(n_matching_het)/float(n_compare)
part_non_matching_het =
float(n_non_matching_het)/float(n_compare)
part_hom_ref = float(n_hom_ref)/float(n_compare)
part_hom_alt = float(n_hom_alt)/float(n_compare)
part_not_called=float(n_not_called)/float(n_compare)
part_not_accouted=float(n_not_accouted)/float(n_compare)

part_not_in_compare=float(n_not_in_compare)/float(n_comp
are)
#How many records were missing from the compare vcf
(which were present in the master vcf)
n_master_minus_n_compare=n_master - n_compare
part_not_found=float(n_master_minus_n_compare)/float(n_m
aster)

count_num_compare=n_matching_het+n_non_matching_het+n_ho
m_ref+n_hom_alt+n_not_called+n_not_accouted
check_num_compare=float(count_num_compare)/float(n_compa
re)
check_parts_compare=part_non_matching_het+part_het+part_
hom_ref+part_hom_alt+part_not_called+part_not_accouted+p
art_not_in_compare
sys.stderr.write("Number of sites in master "+
str(n_master)+ "\n")
sys.stderr.write("Number of sites in compare "+
str(n_compare)+ "\n")
sys.stderr.write("No. missing (not-called) sites: "+
str(n_master_minus_n_compare)+ "\n")
sys.stderr.write("Number of sites allocated a category:
" + str(count_num_compare)+ "\n")

#print("Number + part of non-matching heterozygous sites
" + str(n_non_matching_het) + "(" +
str(part_non_matching_het) + ")" )
#print("Number + part of matching heterozygous sites " +
str(n_matching_het) + "(" + str(part_het) + ")" )
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
#print("Number + part of matching homozygous ref sites "
+ str(n_hom_ref) + "(" + str(part_hom_ref) + ")" )
#print("Number + part of matching homozygous alt sites "
+ str(n_hom_alt) + "(" + str(part_hom_alt) + ")" )

# print("Number + part of sites not called " +
str(n_not_called) + "(" + str(part_not_called) + ")" )
# print("Number + part of genotypes not accounted for "
+ str(n_not_accouted) + "(" + str(part_not_accouted) +
")" )
# print("Number + part of sites not found " +
str(n_not_in_compare) + "(" + str(part_not_in_compare) +
")" )
sys.stderr.write("Checking that number of called
genotypes add up to number of records. This number
should be one: " + str(check_num_compare)+ "\n" )
sys.stderr.write("Checking that parts of called
genotypes add up to number of records. This number
should be very close to one: " +
str(float(check_parts_compare))+ "\n")


print("sample,n_not_called,part_not_called,n_non_matchin
g_het,part_non_matching_het,n_matching_het,part_het,n_ho
m_ref,part_hom_ref,n_hom_alt,part_hom_alt")
print(sample_name+","+str(n_not_called)+","+str(part_not
_called)+","+str(n_non_matching_het)+","+str(part_non_ma
tching_het)+","+str(n_matching_het)+","+str(part_het)+",
"+str(n_hom_ref)+","+str(part_hom_ref)+","+str(n_hom_alt
)+","+str(part_hom_alt))
```

The output from this is a comma-delimited file providing information such as count and proportion of homozygous ref and alt, and heterozygous alleles. I created a combined file as follows:

```
head -n 1 slurm.27401190.out > allocated_genotypes.txt
for f in slurm.*.out; do tail -n 1 $f >>
allocated_genotypes.txt ; done
```

3.3 False discovery rate

Now that we have the list of fixed alleles in ferrets vs polecats, we can test the false discovery rate by looking at these snps in the 12 Mb Broad Sequence array.

Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

The Broad array was mapped to the Broad reference and I'm using the Bionano reference.

Create a new reference sequence

I extracted the 6 x 2Mb fasta sequences from the Broad reference, blasted it against the Bionano reference and identified the sequence array co-ordinates on the Bionano reference.

Get the fasta sequences for the array:

```
source bedtools-2.28.0
bedtools getfasta -fi  MusPutFur1.0_bionano.fasta -fo
MusPutFur1.0_bionano_seq_array.fasta -bed
MusPutFur1.0_bionano_seq_array.intervals
```

I also need to create a version of the dom_ep_fixed.vcf file that only contains the variants within the sequence array.

```
source gatk-4.1.3.0
gatk  SelectVariants -V dom_ep_fixed.vcf -L
bionano_broad_array_coords.list   -O
dom_ep_fixed_broad_array.vcf.gz -R
../../../reference/MusPutFur1.0_bionano.fasta
```

Then, use the output fastq files to map to the Bionano sequence array, and output SNPs within the sequence array co-ordinates.

```
#R1 and R2 refer to the forward and reverse reads
R1=$1
R2=$2

fpath="$(basename $R1)"
samplename="$(cut -d '.' -f1 <<< $fpath)"
bamname=./bams/$samplename
vcfname=./vcfs/$samplename.g.vcf.gz
echo $samplename
echo $fname
source bwa-0.7.7
source samtools-1.7
source jre-8u92
source gatk-4.1.3.0
srun bwa mem -t 8 -M
MusPutFur1.0_bionano_seq_array.fasta $R1 $R2 | samtools
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
sort -@ 8 -o $bamname.bam -
srun  java -jar -XX:ParallelGCThreads=2 -Xmx50G
/ei/software/testing/picardtools/2.21.4/x86_64/bin/picar
d.jar  MarkDuplicates I=$bamname.bam
O=$bamname\_mkdups.bam  M=./metrics/$samplename.metrics
srun java -jar  -XX:ParallelGCThreads=2 -Xmx50G
/ei/software/testing/picardtools/2.21.4/x86_64/bin/picar
d.jar  AddOrReplaceReadGroups I=$bamname\_mkdups.bam
O=$bamname\_rg.bam  RGID=$samplename RGLB=lib1
RGPL=illumina RGSM=$samplename  RGPU=$samplename
srun samtools index $bamname\_rg.bam
srun gatk  --java-options "-Xmx50G -
XX:ParallelGCThreads=14" HaplotypeCaller  -R
MusPutFur1.0_bionano_seq_array.fasta -I $bamname\_rg.bam
-O $vcfname  -ERC BP_RESOLUTION -L
bionano_broad_array_coords.list
```

Then create a GATK genomicsDB, joint genotype all the samples and extract sites in the fixed homozygous ref/var dom_ep_fiexed.vcf file

```
source jre-8u92
source gatk-4.1.3.0
#create the GenomicsDB
srun  gatk --java-options "-Xmx150g -
XX:ParallelGCThreads=6"  GenomicsDBImport  --tmp-
dir=/ei/scratch/ethering/tmp/ -R
MusPutFur1.0_bionano_seq_array.fasta --intervals
MusPutFur1.0_bionano_seq_array.intervals --genomicsdb-
workspace-path  broad_array_GenomicsDB --sample-name-map
gvcf_sample.map  -max-num-intervals-to-import-in-
parallel 6 --merge-input-intervals  --overwrite-
existing-genomicsdb-workspace

#create an 'all-sites' VCF file
srun  gatk --java-options "-Xmx150g -
XX:ParallelGCThreads=6" GenotypeGVCFs -R
MusPutFur1.0_bionano_seq_array.fasta -V
gendb://broad_array_GenomicsDB  --new-qual -O
broad_array_genotyped.vcf.gz-all-sites

#Extract the sites that are in the fixed sites VCF file.
srun  gatk --java-options "-Xmx150g -
XX:ParallelGCThreads=6" SelectVariants   -R
MusPutFur1.0_bionano_seq_array.fasta -V
broad_array_genotyped.vcf.gz  -O
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
broad_array_fixed.vcf.gz -L
dom_ep_fixed_broad_array.vcf.gz  -xl-select-type INDEL
```

The output from this is a VCF file that contains all 76 samples from the Broad array, covering each site that is called as fixed in the dom_ep_fixed_broad_array.vcf.gz file (250 sites).

Next I want to see how many sites, along with how many samples are not called as homozygous ref.

```
import vcf
import sys


vcf_file = sys.argv[1]
min_samples = sys.argv[2]

vcf_reader = vcf.VCFReader(filename=vcf_file)

n_sites = 0
n_ref_sites = 0
n_alt_sites = 0
n_not_called = 0

n_called_genotypes = 0
n_unknown_genotypes = 0
n_ref_genotypes= 0
n_alt_genotypes = 0
n_het_genotypes = 0

allele_freq = 1;

for record in vcf_reader:
    n_sites+=1
    n_called_genotypes+=(record.num_called)
    n_unknown_genotypes+=(record.num_unknown)
    if record.is_monomorphic and record.num_called >=
int(min_samples):
        af = float(record.num_hom_ref/record.num_called)
        #print(str(record.CHROM),'\t',record.POS,'\t',
record.REF,'\t',record.ALT,
'\t',record.num_called,'\t',record.num_hom_ref, '\t',
af, sep='')
        n_ref_sites+=1
        #two ref alleles at each site
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
        n_ref_genotypes+=(record.num_hom_ref)
        if (af < allele_freq):
            allele_freq = af
    elif record.is_snp and record.num_called
>=int(min_samples):#no indels
        #print("SNP: ", str(record), " ",
record.var_type)
        af = record.num_hom_alt/record.num_called
        #print(str(record.CHROM),'\t',record.POS, '\t',
record.REF,'\t',record.ALT,
'\t',record.num_called,'\t',record.num_hom_alt, '\t',
af, sep='')
        n_alt_sites+=1
        #2 alt alleles at each site
        n_alt_genotypes+=(record.num_hom_alt)
        n_het_genotypes+=(record.num_het)

    else:
        print("Not called: ", str(record), " ",
record.var_type)
        n_not_called+=1

sys.stdout.write("No. sites ")
sys.stdout.write(str(n_sites))
sys.stdout.write("\n")
sys.stdout.write("No. hom ref sites ")
sys.stdout.write(str(n_ref_sites))
sys.stdout.write("\n")
sys.stdout.write("No. hom alt sites ")
sys.stdout.write(str(n_alt_sites))
sys.stdout.write("\n")
sys.stdout.write("No. sites not called ")
sys.stdout.write(str(n_not_called))
sys.stdout.write("\n")

site_fdr = float(n_alt_sites/(n_sites))
sys.stdout.write("Site FDR ")
sys.stdout.write(str(site_fdr))
sys.stdout.write("\n")

sys.stdout.write("No. called genotypes ")
sys.stdout.write(str(n_called_genotypes))
sys.stdout.write("\n")
sys.stdout.write("No. hom ref genotypes ")
sys.stdout.write(str(n_ref_genotypes))
sys.stdout.write("\n")
sys.stdout.write("No. hom alt genotypes ")
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn

```
sys.stdout.write(str(n_alt_genotypes))
sys.stdout.write("\n")
sys.stdout.write("No. het genotypes ")
sys.stdout.write(str(n_het_genotypes))
sys.stdout.write("\n")
sys.stdout.write("No. unknown genotypes (e.g. not
called) ")
sys.stdout.write(str(n_unknown_genotypes))
sys.stdout.write("\n")
genotype_fdr =
((float(n_alt_genotypes)+float(n_het_genotypes))/float(n
_called_genotypes))
sys.stdout.write("Allele FDR ")
sys.stdout.write(str(genotype_fdr))
sys.stdout.write("\n")

sys.stdout.write("Lowest allele frequency = ")
sys.stdout.write(str(allele_freq))
sys.stdout.write("\n")
```


Citation: Graham Etherington Introgression https://dx.doi.org/10.17504/protocols.io.bq7tmznn