Methods - Read filtering, mapping, genome coverage, and variant calling
Raw reads were filtered in BBmap in the BBTools package. The BBmap settings were to trim all known Illumina adapter sequences. Reads with more than 0 Ns were discarded.
Reads shorter than 41 bases after trimming were discarded. Read pairs were discarded only if both were shorter than 0.33 of the original length after trimming. Reads were compared to 'human', 'cat', 'dog', 'mouse', and 'microbes' databases to remove potential contaminations.

BBmap command:
```
rqcfilter.sh usejni in=infile1.R1.fastq.gz
in2=infile2.R2.fastq.gz path=null rna=f trimfragadapter=t
qtrim=r trimq=0 maxns=0 maq=13 minlen=41 mlf=0.33 phix=t
removehuman=t removedog=t removecat=t removemouse=t khist=t
removemicrobes=t sketch clumpify=f tmpdir= usejni=f
```

To obtain mapped genomes to use for SNP-calling, for each individual sample: we mapped the reads, and then sorted, marked and deleted PCR duplicates, indexed, and calculated their genome coverage.

To map our cleaned, filtered reads, we used v3 of the *Suillus luteus* as our reference made of 67 scaffolds. In Bowtie2, we indexed the reference genome at first, and then we mapped the clean reads of each individual onto it.

Bowtie2 index reference command:
```
bowtie2-build reference.fa suiluRef
```

Bowtie2 map samples to reference command:
```
bowtie2 -x suiluRef --interleaved samplefile.fastq  -S
outsamplefile.map.sam
```

We sorted the mapped alignments in picard by genomic coordinate. We added read groups in picard based on read and illumina run information we obtained from JGI, (see table).

Picard sort command:
```
java -jar picard.jar SortSam INPUT=outsamplefile.map.sam
OUTPUT=outsamplefile.sort.sam SORT_ORDER=coordinate
TMP_DIR=/mnt/lustrefs/work/anna.bazzicalupo/3_sort/
```
("TMP_DIR=" was used as our server had a 2GB temporary directory)

We converted the sam files to bam files, as they are smaller and easier to handle.

SamTools sam to bam format command:
```
samtools view -bS -o outsamplefile.sort.bam
outsamplefile.sort.sam
```

For all subsequent analyses we needed to add information to the reads (‘read group’ code, Illumina run code, Illumina barcodes, matched with the sample ID), we did that in Picard.

Picard add read group command (example codes):
```
java -jar picard.jar AddOrReplaceReadGroups I=
outsamplefile.sort.bam O outsamplefile.RG.bam
RGID=12232.1.246284.AGCTAAC-GGTTAGC RGLB=BXUNY RGPL=illumina
RGPU=AGCTAAC-GGTTAGC RGSM=502931_1151568
```
(codes for our samples are in Supplementary Table 5)

We marked and deleted PCR duplicates in Picard.

Picard delete PCR duplicate command:
```
java -jar picard.jar MarkDuplicatesWithMateCigar
REMOVE_DUPLICATES=true INPUT=outsamplefile.RG.bam
OUTPUT=outsamplefile.mark.bam M=metrics_outsamplefile.txt
```

We created an index of each bam file in SamTools.

SamTools index command:
```
samtools index outsamplefile.mark.bam
```

We calculated average coverage per genome using the depth of coverage tool in GATK.

GATK coverage command:
```
java -jar GenomeAnalysisTK.jar -T DepthOfCoverage -R
reference.fa -o outsamplefile_cov -I outsamplefile.mark.bam
```

We called variants in GATK. A vcf file was obtained for each sample by running GATK HaplotypeCaller in gvfc mode.

GATK HaplotypeCaller gvcf mode command:
```
java -jar GenomeAnalysisTK.jar -R reference.fa -T
HaplotypeCaller -I outsamplefile.mark.bam --emitRefConfidence
GVCF -o outsamplefile.g.vcf
```

All samples were genotyped together and SNPs were called in GATK using GenotypeGVCF tool.

GATK HaplotypeCaller gvcf mode command:

```
java -jar $ GenomeAnalysisTK.jar -R reference.fa -T
GenotypeGVCFs --variant outsamplefile1.g.vcf --variant
outsamplefile2.g.vcf -o gvcf.snps.indels.2019-01-30dik.vcf
```

We applied filters to the vcf file to remove poor quality SNPs. In GATK, we removed all non-biallelic SNPs.

Filter non-biallelic SNPs command:

```
java -jar GenomeAnalysisTK.jar -T SelectVariants -R reference.fa
-V gvcf.snps.indels.2019-01-30dik.vcf -o gvcf.biallelic.2019-01-
30dik.vcf --selectTypeToInclude SNP --restrictAllelesTo
BIALLELIC
```

In VCFTools we removed all SNPs with more than 200x or less than 5x coverage and with minimum quality above 30. This is the file we used for all subsequent analyses. To explore the effect of stricter filtering on the SNPs we also filtered the vcf file by removing all SNPs with more than 200x or less than 10x coverage.

Filter SNPs coverage commands:

```
vcftools --vcf gvcf.biallelic.2019-01-30dik.vcf --remove-
filtered-all --minQ 30 --min-meanDP 5 --max-meanDP 200 --out
gvcf.filtered.2019-01-30dik –recode
```

Methods - Population structure and summary statistics

We performed a structure analysis using fastSTRUCTURE, we ran the analysis 5 times specifying K as 1, 2, 4, 6, and 38.

We converted the vcf file to .ped and .map in VCFTools:

```
vcftools --vcf gvcf.filtered.2019-01-30dik.recode.vcf --plink
```

FastSTRUCTURE requires a .bed file format. We used plink to get a .bed format file, command:

```
plink --file gvcf.filtered.2019-01-30dik --make-bed --out
gvcf.filtered.2019-01-30dik
```

We implemented Python fastSTRUCTURE (ran K=1,2,4,6 and 38) to estimate the number of populations for our samples:

```
python structure.py -K 1 –input=gvcf.filtered.2019-01-30dik --
output=gvcf.filtered.2019-01-30dikStructure --full --seed=100
```

We calculated summary statistics π and Tajima's D in the sci-kit allel Python package:

Methods – measures of allelic divergence

We calculated F_{ST} and d_{XY}.

We calculated F_{ST} in VCFTools:

```
vcftools --vcf gvcf.filtered.2019-01-30dik.recode.vcf --fst-
window-size 5000 --weir-fst-pop samplelist.popdik1.txt --weir-
fst-pop samplelist.popdik2.txt --out 2019-01-315kbFSTdik
```

To calculate dXY we used in house scripts requiring a tab file, so we used VCFTools vcf to tab command:

```
cat gvcf.filtered.2019-01-30dik.recode.vcf | vcf-to-tab >
gvcf.2019-01-30dik.recode.tab
```

To calculate d_{XY} we used in-house scripts written in Python using the egglib package:
To reformat tab file to fasta:

```
make_fasta_from_ref_and_tab_diploid_wN.py
```

Make fasta for scaffolds with no SNPs:

```
make_fasta_for_scaffolds_wo_SNPs_Sb_corrected.py
```

Calculate d_{XY} in the egglib Python package:

```
egglib_compute_Xkb_windows_scaffolds.py
Da_compute_Xkb_windows_fasta_byscaffold.py
```

Methods - Estimates of copy number variation

We used software ControlFreeC to estimate copy-numbers for individual genomes. ControlFreeC requires a mpileup file and a 'config' file.

We made an mpileup file in SamTools for each genome:

```
samtools mpileup -f reference.fa outsamplefile.mark.bam >
cnv.outsamplefile.pileup
```

We made a 'config' file for each individual, below is an example content of the Config files used as input for ControlFreeC:

```
[general]
chrLenFile = suilu_chr.len
ploidy = 2
window = 250
breakPointThreshold = 0.8
coefficientOfVariation = 0.062
minExpectedGC = .35
maxExpectedGC = .55

[sample]
mateFile = outsamplefile.pileup
inputFormat = pileup
mateOrientation=FR
```

Then we ran ControlFreeC.

```
~/FREEC-11.5/src/freec -conf config_outsamplefile1.txt
```