Supplementary Material: Code Documentation

The discrepancy between the high population size and the low genetic diversity in European grey seals

Theresa Koller
Master's Thesis in Ecology and Evolutionary Biology
University of Helsinki
1st supervisor: Dr. Ari Löytynoja
2nd supervisor: Dr. Morten Tange Olsen

2025-06-24

This document illustrates the data processing and analysis pipeline used in my thesis project. For further information visit the main document. The headline numbering here matches that in the main document.

The code illustration in this document, as well as the methodology itself, are highly inspired by Olkkonen & Löytynoja (2023) and the material of the course EEB-210 at the University of Helsinki offered by Dr. Ari Löytynoja (A.L.) in 10-12/2024.

All data processing and analysis work was performed on *puhti* from CSC - IT Center of Science, Finland. Large data was stored on *Allas* from CSC.

The `\` is used to omit the following "new line" character. Thus, code that is separated by `\` should be run as a single line.

2. Materials and Methods

2.1. Data

The data consisted of 140 whole-genome sequences from different seal species, grey seals (*Halichoerus grypus*) and ringed seals (*Pusa hispida*), as well as Saimaa ringed seals (*Pusa saimensis*), originating from two different data sources (CPH and HEL).

2.2 Data processing

Set up

In this document, all the steps are performed in the same working directory (`$work`). For simplicity this documentation assumes that all tools are available in the bin directory. In reality many of the tools used were available on *puhti*, which were loaded with `module load tool`. The rest of the tools were made available by A.L.

```
work=$(pwd)
tools=$work/bin
bwa=$tools/bwa
samtools=$tools/samtools
```

```
gatk3=$tools/gatk3
gatk=$tools/gatk
bcftools=$tools/bcftools
rmask=$tools/RepeatMasker
seqbility=$tools/seqbility
bedtools=$tools/bedtools
plink=$tools/plink
vcftools=$tools/vcftools
miniprot=$tools/miniprot
busco=$tools/busco
beagle=$tools/beagle
bedextract=$tools/bedextract
genmulti=$tools/generate_multihetsep.py
msmc2=$tools/msmc2
combinecc=$tools/combineCrossCoal.py
msmc_im=$tools/MSMC_IM.py
```

Note: `seqbility` contains `splitfa`, `gen_raw_mask.pl`, `gen_mask`, and `makeMappabilityMask.py`.

Preparation of the reference genome

The ribbon seal reference genome was available in *FASTA* format. The reference genome was unzipped and indexed.

```
ref=$work/reference/ribbon
gunzip $ref.fa.gz
$samtools faidx $ref.fa
$samtools dict $ref.fa > ref.dict
$bwa index $ref.fa
```

Read mapping

Here and in the following steps (**Realignment around indels** and **Variant calling**) the pipeline is illustrated for one sample (`smp1`); in my thesis these steps were performed for all 140 grey and ringed seal samples.

The grey and ringed seal *FASTQ* reads were mapped to the reference genome resulting in *BAM* files. `fastq` and `bams` were the directories, where the reads and mapped files were stored, respectively. Mapping required defining a “readgroup” (`rgroup`) (Further information: https://gatk.broadinstitute.org/hc/en-us/articles/360035890671-Read-groups).

```
mkdir $work/fastq
fastq=$work/fastq
mkdir $work/bams
bams=$work/bams
rgroup="@RG\tID:sample-1\tLB:Lib\tSM:sample-1\tPL:Plfm"
smp=smp1
bwa mem -R ${rgroup} $ref.fa $fastq/${smp}_[12].fq.gz \
| $samtools fixmate -m - -  \
| $samtools sort - -o $bams/${smp}_bwa.bam \
&& $samtools index $bams/${smp}_bwa.bam
```

Realignment around indels

A list of regions to realign was created and the *BAM* files were realigned around the target. The duplicates were marked and the resulting file was indexed.

```
$gatk3 -T RealignerTargetCreator \
-R ${ref} -I $bams/${smp}_bwa.bam -o $bams/$smp.intervals
$gatk3 -T IndelRealigner \
-R ${ref} -I $bams/${smp}_bwa.bam -targetIntervals $bams/$smp.intervals \
-o $bams/${smp}_real.bam
$samtools markdup $bams/${smp}_real.bam $bams/${smp}_mdup.bam
$samtools index $bams/${smp}_mdup.bam
```

Variant calling

Variants were called and the *GVCF* files were stored in the `gvcfs` directory.

```
mkdir $work/gvcfs
gvcfs=$work/gvcfs
gatk HaplotypeCaller \
    -R $ref.fa -I $bams/${smp}_mdup.bam \
    -O $gvcfs/$smp.gvcf.gz -ERC GVCF
```

Genotype calling

Assuming the last three steps were performed for all samples, there should now be 140 *GVCF* files in the `gvcfs` directory. For simplicity, the next step is illustrated with only 8 samples, i.e. two HEL grey seals `HEL_grey_1.gvcf.gz` & `HEL_grey_2.gvcf.gz`, two CPH grey seals `CPH_grey_1.gvcf.gz` & `CPH_grey_2.gvcf.gz`, two HEL ringed seals `HEL_ringed_1.gvcf.gz` & `HEL_ringed_2.gvcf.gz`, two CPH ringed seals `CPH_ringed_1.gvcf.gz` & `CPH_ringed_2.gvcf.gz`.

In reality, **genotype calling**, **joint calling**, and **SNP filtering** was done in 19 batches, for which the reference genotype was split into 19 parts with *AWK*. These batches were then called separately by specifying the respective reference genome batch with the `-O` flag. The batches were then concatenated using `bcftools concat`.

For simplicity, this documentation illustrates the processing of the entire genome (rather than the batches). The *GVCF* files were combined in three groups resulting in three combined *GVCF* files in the `gvcf` directory: (i) all grey seal samples (HEL and CPH) `grey.gvcf.gz`, (ii) HEL ringed seals `ringed_HEL.gvcf.gz`, (iii) CPH ringed seals `ringed_CPH.gvcf.gz`. The CPH and HEL ringed seals were called separately because the CPH ringed seals were added later.

```
mkdir $work/gvcf
gvcf=$work/gvcf

# all grey seals
smp_HEL_grey_1=$gvcfs/HEL_grey_1.gvcf.gz
smp_HEL_grey_2=$gvcfs/HEL_grey_2.gvcf.gz
smp_CPH_grey_1=$gvcfs/CPH_grey_1.gvcf.gz
smp_CPH_grey_2=$gvcfs/CPH_grey_2.gvcf.gz
$gatk CombineGVCFs \
 -R $ref.fa \
 -V $smp_HEL_grey_1 \
 -V $smp_HEL_grey_2 \
 -V $smp_CPH_grey_1 \
 -V $smp_CPH_grey_2 \
 -O $gvcf/grey.gvcf.gz

# HEL ringed seals
smp_HEL_ringed_1=$gvcfs/HEL_ringed_1.gvcf.gz
smp_HEL_ringed_2=$gvcfs/HEL_ringed_2.gvcf.gz
$gatk CombineGVCFs \
 -R $ref.fa \
 -V $smp_HEL_ringed_1 \
 -V $smp_HEL_ringed_2 \
 -O $gvcf/ringed_HEL.gvcf.gz

# CPH ringed seals
smp_CPH_ringed_1=$gvcfs/CPH_ringed_1.gvcf.gz
smp_CPH_ringed_2=$gvcfs/CPH_ringed_2.gvcf.gz
$gatk CombineGVCFs \
 -R $ref.fa \
 -V $smp_CPH_ringed_1 \
 -V $smp_CPH_ringed_2 \
 -O $gvcf/ringed_CPH.gvcf.gz
```

Joint calling

As a next step each of the three combined *GVCF* files was joint called resulting in three *VCF* files stored in the `vcf` directory.

```
mkdir $work/vcf
vcf=$work/vcf

# all grey seals
$gatk --java-options "-Xmx4g" GenotypeGVCFs \
 -R $ref.fa \
 -V $gvcf/grey.gvcf.gz \
 -O $vcf/grey.vcf.gz

# HEL ringed seals
$gatk --java-options "-Xmx4g" GenotypeGVCFs \
 -R $ref.fa \
 -V $gvcf/ringed_HEL.gvcf.gz \
 -O $vcf/ringed_HEL.vcf.gz

# CPH ringed seals
$gatk --java-options "-Xmx4g" GenotypeGVCFs \
 -R $ref.fa \
 -V $gvcf/ringed_CPH.gvcf.gz \
 -O $vcf/ringed_CPH.vcf.gz
```

SNP filtering

The *VCF* files were then filtered to remove all non-binary and fixed SNPs.

```
$bcftools view -v snps -m2 -M2 -q0.001:minor $vcf/grey.vcf.gz \
> $vcf/grey_snp.vcf.gz
$bcftools view -v snps -m2 -M2 -q0.001:minor $vcf/ringed_HEL.vcf.gz \
> $vcf/ringed_HEL_snp.vcf.gz
$bcftools view -v snps -m2 -M2 -q0.001:minor $vcf/ringed_CPH.vcf.gz \
> $vcf/ringed_CPH_snp.vcf.gz
```

Joint masking

The repetitive elements in the reference genome were identified using `RepeatMasker` and dog repeat libraries resulting in the *GFF* file `$ref.fa.out.gff`

```
mkdir $work/RM
RM=$work/RM
ncpu=40
$rmask -xsmall -gff --engine rmblast -dir $RM -pa $ncpu \
-species "Canis familiaris" $ref.fa
```

To create the positive mask, the reference genome was split into 75 bp fragments with `splitfa` and realign back with `bwa` to the reference genome. Based on that a *FASTA* sequence was created with `gen_raw_mask.pl` and `gen_mask`. The *FASTA* sequence was then converted into *BED* format using `makeMappabilityMask.py`.

```
mkdir $work/maskdir
maskdir=$work/maskdir
cd $maskdir
$seqbility/splitfa $refs.fa 75 | split -l 20000000
for i in $(ls x??);
  do  $bwa aln -R 1000000 -O 3 -E 3 $refs.fa $i > $i.sai
done
for i in $(ls x??);
  do  $bwa samse $refs.fa $i.sai $i > $i.sam
done
cat x??.sam | $seqbility/gen_raw_mask.pl > rawMask.fa
$seqbility/gen_mask -l 35 -r 0.5 rawMask.fa > mask.fa
python $seqbility/makeMappabilityMask.py
out=$maskdir/posmask.bed.gz
for i in $(ls $maskdir/posmask);
  do zcat $maskdir/posmask/$i | bgzip -c >> $out
done
```

Repeat mask and positive mask were then combined into a joint mask, excluding the repeat regions and including the positive mask.

```
$bedtools subtract -a $maskdir/posmask.bed -b $RM/$ref.fa.out.gff \
> $maskdir/joint_mask.bed
```

Then the grey and ringed seal *VCF* files were filtered based on the joint mask.

```
mask=$maskdir/joint_mask.bed
$bcftools view -T $mask $vcf/grey_snp.vcf.gz -Oz -o $vcf/grey_snp_mask.vcf.gz
$bcftools view -T $mask $vcf/ringed_HEL_snp.vcf.gz -Oz -o $vcf/ringed_HEL_snp_mask.vcf.gz
$bcftools view -T $mask $vcf/ringed_CPH_snp.vcf.gz -Oz -o $vcf/ringed_CPH_snp_mask.vcf.gz
```

The length of the whole genome, and the joint mask was determined with *AWK*, using the reference genome index and the mask *BED* file, respectively.

```
awk '{s+=$2}END{print s}' $ref.fa.fai > ${ref}_length.txt
awk '{s+=$3-$2}END{print s}' $mask > $maskdir/joint_mask_length.txt
```

Merge CPH and HEL ringed seals

Next, the HEL and CPH ringed seal data was merged.

```
$bcftools merge $vcf/ringed_HEL_snp_mask.vcf.gz $vcf/ringed_CPH_snp_mask.vcf.gz \
-Oz -o $vcf/ringed_snp_mask.vcf.gz
```

Sequencing depth

The sequencing depth of the first ten million sites was estimated with `samtools depth` for all grey seals and HEL ringed seals, and with `bcftools query` for CPH ringed seals. The `samtools` approach was done for all samples individually and is here illustrated for one sample `smp1`. The `bcftools` is performed a *VCF* file and outputs the estimated sequencing depth for each samples.

```
smp=smp1
$samtools depth -a ${smp}_markdup.bam | head -10000000 | awk '{s+=\$3}END{print s/10000000}'

bcftools query -f '[%DP\t]\n' $vcf/ringed_CPH_snp_mask.vcf.gz -H | head -10000000 \
| sed 's/#//g;s/:DP//g;s/\[[0-9]*\]//g' \
| awk '{if(NR==1){
        for(i=1;i<=NF;i+=1){ smp[i]=$i }
        } else {
            for(i=1;i<=NF;i+=1){ dpt[i]+=$i }
            }
        }
        END{
            for(i=1;i<=NF;i+=1){ print smp[i],dpt[i]/NR }
        }'
```

2.3. Population Structure

PCA and IBS distance analysis were performed with `plink`. PCA was performed for all grey seals, CPH Atlantic, CPH Baltic grey seals, and and all ringed seals. The output of the PCA is an *eigenvec* and an *eigenval* file. IBS distances were derived for CPH grey seals and HEL ringed seals, outputting a *mdist.id* and a *mdist.gz* file. The respective output files were visualized in R. The respective subgroups were filtered from the *VCF* files using `bcftools view` and referring to the `metadata.tsv`, that holds the sample name in the first row.

PCA

```
mkdir $work/explore
explore=$work/explore

# all grey seals
$plink --vcf $vcf/grey_snp_mask.vcf.gz --pca --allow-extra-chr --out $explore/grey
```

```
# CPH Atl grey seals
$bcftools view -S <(grep CPH metadata.tsv | grep Atlantic | cut -f1) \
$vcf/grey_snp_mask.vcf.gz | $plink --vcf - --pca --allow-extra-chr \
--out $explore/grey_CPH_Atl

# CPH Bal grey seals
$bcftools view -S <(grep CPH metadata.tsv | grep Baltic | cut -f1) \
$vcf/grey_snp_mask.vcf.gz | $plink --vcf - --pca --allow-extra-chr \
--out $explore/grey_CPH_Bal

# all ringed seals
$plink --vcf $vcf/ringed_snp_mask.vcf.gz --pca --allow-extra-chr --out $explore/ringed
```

IBS

```
# CPH grey seals
$bcftools view -S <(grep CPH metadata.tsv | cut -f1) $vcf/grey_snp_mask.vcf.gz \
| $plink --vcf - --distance 1-ibs square gz \
--allow-extra-chr --out $explore/grey_CPH

# HEL ringed seals
$plink --vcf $vcf/ringed_HEL_snp_mask.vcf.gz --distance 1-ibs square gz \
--allow-extra-chr --out $explore/ringed_HEL
```

Note: PCA and IBS can also be run in one go by adding both the `--pca` and the `--distance 1-ibs square gz` flag.

2.4. Nucleotide diversity (π) and heterozygosity (H)

Pi and H were calculated with `vcftools --site-pi`. For H, this was done for each sample individually. For π, this was done for a group of samples.

π

The π calculation is illustrated for CPH Atlantic grey seals, the π calculations for other study units were done in the same way.

```
$bcftools view -S <(grep CPH metadata.tsv | grep Atlantic | cut -f1) \
$vcf/grey_snp_mask.vcf.gz | $vcftools --vcf - --site-pi -c --out $Pi_H/Pi_grey_CPH_Atl
numnan=$(cat $Pi_H/Pi_grey_CPH_Atl.sites.pi | tail -n+2 | grep nan | wc -l)
corlength=$(( $length-$numnan ))
cat $Pi_H/Pi_grey_CPH_Atl.sites.pi | tail -n+2 | grep -v nan \
| awk '{s+=$3}END{print s/"'"$corposlen"'"}' > $Pi_H/zPi_grey_CPH_Atl.txt
```

H

This illustrates the H calculation for grey seals, the calculation for ringed seals was done in the same way.

```
mkdir $work/Pi_H
Pi_H=$work/Pi_H

# length of the joint mask
```

```
length=$maskdir/joint_mask_length.txt

# calculate H per site
for smp in $(cut -f1 metadata.tsv); do
        $bcftools view -S $smp $vcf/grey_snp_mask.vcf.gz \
        | $vcftools --vcf - --site-pi -c --out $Pi_H/H_grey_$smp
done

cd $Pi_H

# calculate mean H across the whole genome
samples=$(ls H_grey_*)
for file in $samples; do
        numnan=$(cat $file | tail -n+2 | grep nan | wc -l)
        corlength=$(( $lenth-$numnan ))
        cat $file | tail -n+2 | grep -v nan | awk '{s+=$3}END{print s/"'"$corlength"'"}' \
        > z$file.txt
done

# combine H estimates of all individuals
awk 'FNR==1 {print FILENAME "\t" $0}' zH_grey_* | sort | sed 's/zH_grey_//' \
| sed  's/.sites.pi.txt//' > H_grey.tsv
```

The averaged H per study unit was calculated in R.

2.5. Nucleotide diversity (π) and heterozygosity (H) in genomic windows

For that the .sites.pi files from 2.4. were used. The π and H in windows were calculated in R. The script was heavily inspired by Olkkonen & Löytynoja (2023). This documentation refers to a source script window_function.R found in the same GitHub repository. Make sure to specify the correct joint mask and reference index in the source script.

This documentation illustrates the calculations for CPH Atlantic grey seals.

π

```
setwd("work") # set the working directory accordingly
source("window_function.R") # get the functions "windows" and "bins.plot"

# read in data
Pi_grey_CPH_Atl <-
  as.data.frame(read.table("Pi_H/Pi_grey_CPH_Atl.sites.pi", sep = "\t", header = T))
# calculate windows
Pi_grey_CPH_Atl_windows <- windows(Pi_grey_CPH_Atl)
# delete to save memory
rm(Pi_grey_CPH_Atl)
# counts the windows in 0.0001 pi bins
Pi_grey_CPH_Atl_plot <- bins.plot(Pi_grey_CPH_Atl_windows)
bin <- c(seq(0,0.00519,0.0001),0.025)
# add other study units here if needed
Pi <- as.data.frame(cbind(bin, Pi_grey_CPH_Atl_plot))
colnames(Pi) <- c("bins", "Pi_grey_CPH_Atl")
# combine
```

```
Pi.long <- gather(Pi, study_units, count, -bins)
# Pi.long can then be plotted with x=bins and y=count
```

H

```
# select the correct samples here
H_grey_CPH_Atl_files <-
  c("H_grey_smp1.sites.pi", "H_grey_smp2.sites.pi", "H_grey_smp3.sites.pi")
H_grey_CPH_Atl_windows <- list()
a <- 1

for (i in H_grey_CPH_Atl_files) {
  filename <- paste0("Pi_H/", i)
  # read in data
  data <- as.data.frame(read.table(filename, sep = "\t", header = T))
  # calculate windows
  H_grey_CPH_Atl_windows[[a]] <- windows(data)
  # delete to save memory
  rm(data)
  print(i)
  a <- a+1
}

H_grey_CPH_Atl_names1 <- gsub("^H_grey_", "", H_grey_CPH_Atl_files)
H_grey_CPH_Atl_names <- gsub("^.sites.pi", "", H_grey_CPH_Atl_names1)
H_grey_CPH_Atl_bins <- bin

for (a in 1:length(H_grey_CPH_Atl_windows)) {
  # counts the windows in 0.0001 pi bins
  temp.bins <- bins.plot(H_grey_CPH_Atl_windows[[a]])
  H_grey_CPH_Atl_bins <- cbind(H_grey_CPH_Atl_bins, temp.bins)
}

colnames(H_grey_CPH_Atl_bins) <- c("bins", H_grey_CPH_Atl_names)
# add other study units here if needed
H <- as.data.frame(cbind(H_grey_CPH_Atl_bins, study_unit[,-1]))
# combine
H.long <- gather(H, smp, count, -bins)
# H.long can then be plotted with x=bins and y=count
```

2.6. Nucleotide diversity (π) in genes

CDS

To be able to determine coding site (CDS) regions in grey and ringed seals, the reference genome was annotated. This was done with the northern elephant seal (NES) protein sequence using `miniprot`.

First, the NES protein sequence was downloaded from NCBI.

```
mkdir $work/annotation
annotation=$work/annotation

cd $annotation
```

```
wget https://ftp.ncbi.nlm.nih.gov/genomes/all/annotation_releases/9716/\
GCF_029215605.1-RS_2024_04/GCF_029215605.1_mMirAng1.0.hap1_protein.faa.gz
mv GCF_029215605.1_mMirAng1.0.hap1_protein.faa.gz elephant_seal_protein.faa.gz
cd ..
```

Second, the reference genome was annotated.

```
protein=$annotation/elephant_seal_protein.faa.gz
$miniprot -t8 --gff $ref.fa.gz $protein > \
$annotatation/annotation_elephant_seal.gff
```

Third, CDS regions were selected and the *GFF* files was converted into a *TAB* and a *BED* file.

```
awk '$3 == "CDS" && !/^#/{OFS="\t"; print $1, $4, $5}' \
$annotatation/annotation_elephant_seal.gff | $bedtools sort -i /dev/stdin \
| $bedtools  -i /dev/stdin > $annotatation/elephant_seal_CDS.tab

awk '{OFS="\t";print $1,$2-1,$3}' $annotatation/elephant_seal_CDS.tab \
> $annotatation/elephant_seal_CDS.bed

# determine the length
$bedtools intersect -a $mask -b $annotatation/elephant_seal_CDS.bed \
| awk '{s+=$3-$2}END{print s > "$annotatation/length_CDS.txt"}'
```

BUSCO genes

Carnivore BUSCO genes were found in the reference genome with `busco` and the intersect between the CDS regions and the BUSCO regions was taken with `bedtools`.

```
$busco -c 20 -i $ref.fa -m genome -l carnivora -f > $annotation/busco_genes.bed
# take the intersect with CDS regions
$bedtools intersect -a $annotation/elephant_seal_CDS.bed -b $annotation/busco_genes.bed \
> $annotation/elephant_seal_BUSCO.bed
# determine the length
$bedtools intersect -a $mask -b $annotation/elephant_seal_BUSCO.bed \
| awk '{s+=$3-$2}END{print s > "$annotation/length_BUSCO.txt"}'
```

Immune genes

Immune genes from the ferret were identified by Dr. Mia Valtonen and Dr. Outi Hallikas from the Ensemble database as `$annotation/ferret_immune.txt`. To be able to use those, the reference genome was annotated with the ferret protein sequence from Ensemble.

First, the ferret protein sequence from NCBI was downloaded.

```
cd $annotation
wget https://ftp.ensembl.org/pub/release-113/fasta/\
mustela_putorius_furo/pep/Mustela_putorius_furo.MusPutFur1.0.pep.all.fa.gz
mv Mustela_putorius_furo.MusPutFur1.0.pep.all.fa.gz ferret_protein.fa.gz
cd ..
```

Second, the reference genome was annotated.

```
protein=$annotation/ferret_protein.fa.gz
$miniprot -t8 --gff $ref.fa.gz $protein > $annotation/annotation_ferret.gff
```

Third, the immune regions in the ferret annotation were identified with `grep``` andawk"'.

```
# correct the protein names
sed 's/ENSMPUG/ENSMPUP/g' $annotation/ferret_immune.txt \
> $annotation/ferret_immune_corrected.txt

# select gene immune regions
grep -f $annotation/ferret_immune_corrected.txt $annotation/annotation_ferret.gff \
| awk '$3 == "CDS" && !/^#/{OFS="\t"; print $1, $4, $5}' | $bedtools sort -i /dev/stdin \
| $bedtools  -i /dev/stdin > $annotation/ferret_immune_regions.tab

# convert to BED file
awk '{OFS="\t";print $1,$2-1,$3}' $annotation/ferret_immune_regions.tab \
> $annotation/ferret_immune_regions.bed
```

Forth, overlap the ferret immune genes with the NES CDS regions.

```
$bedtools intersect -a $annotation/elephant_seal_CDS.bed \
-b $annotation/ferret_immune_regions.bed > $annotation/elephant_seal_immune.bed

# determine the length
$bedtools intersect -a $mask -b $annotation/elephant_seal_immune.bed \
| awk '{s+=$3-$2}END{print s > "$annotation/length_immune.txt"}'
```

Filter the data

Filter the *VCF* files based on my annotations. This step is illustrated for grey seal, and was done for ringed seals accordingly.

```
vcf_grey=$vcf/grey_snp_mask.vcf.gz

CDS=$annotation/elephant_seal_CDS.bed
CDS_BUSCO=$annotatio/elephant_seal_BUSCO.bed
CDS_immune=$annotation/elephant_seal_immune.bed

vcf_CDS_grey=$annotation/grey_snp_mask_CDS.vcf.gz
vcf_CDS_BUSCO_grey=$annotatio/grey_snp_mask_CDS_BUSCO.vcf.gz
vcf_CDS_immune_grey=$annotatio/grey_snp_mask_CDS_immune.vcf.gz

$bcftools view -T $CDS $vcf_grey -Oz -o $vcf_CDS_grey
$bcftools view -T $CDS_BUSCO $vcf_grey -Oz -o $vcf_CDS_BUSCO_grey
$bcftools view -T $CDS_immune $vcf_grey -Oz -o $vcf_CDS_immune_grey
```

Then, π was calculated for different study units for the three categories (CDS, BUSCO, and immune), as illustrated in 2.4. The relative π (i.e. π in CDS divided by π in the full genome) was done in R.

2.7. Demographic and isolation-migration inference

For this analysis only CPH data was used. This was performed with `MSMC2`.

Imputation and phasing

MSMC2 requires phased data, imputation was done to improve data quality. This was done for CPH Atlantic grey seals, excluding Russia and Norway, CPH Baltic grey seals and CPH Arctic ringed seals, separately. To achieve those sets the *VCF* files were subsetted with `bcftools view`, as illustrated before leading to the three files: `grey_snp_mask_CPH_Atl.vcf.gz`, `grey_snp_mask_CPH_Bal.vcf.gz`, `ringed_snp_mask_CPH.vcf.gz`.

`beagle` is run twice, once to impute and once to phase. This is illustrated for CPH Atlantic grey seals only, and was done accordingly with the other two sets.

```
mkdir $work/phasing
phasing=$work/phasing

$beagle_tool="java -Xmx145G -jar $beagle"
$beagle nthreads=5 ne=100000 gl=$vcf/grey_snp_mask_CPH_Atl.vcf.gz \
out=$phasing/grey_Atl_imp
$beagle nthreads=5 ne=100000 gt=$phasing/grey_Atl_imp.vcf.gz \
out=$phasing/grey_Atl_phased
```

Baltic and Atlantic grey seal data were then merged with `bcftools merge`.

```
$bcftools index $phasing/grey_Atl_phased.vcf.gz
$bcftools index $phasing/grey_Bal_phased.vcf.gz
$bcftools merge $phasing/grey_Atl_phased.vcf.gz $phasing/grey_Bal_phased.vcf.gz \
-Oz -o  $phasing/grey_phased.vcf.gz
$bcftools index $phasing/grey_phased.vcf.gz
```

This resulted in the two *VCF* files `$phasing/grey_phased.vcf.gz` and `$phasing/ringed_phased.vcf.gz` used for the MSMC2 analysis.

Create *multiHetSep* files

Six samples were chosen, based on sequencing depth, from each study unit to include in the MSMC2 analysis. For the estimation of the sequencing depth see 2.2.

First, the joint mask was split into contigs.

```
mkdir $maskdir/contigs
contigs=$maskdir/contigs

for chr in {001..499}; do
 CHR=Hfa$chr
 $bedextract $CHR $mask > $contigs/$CHR.mask.bed
 gzip $contigs/$CHR.mask.bed
done
```

Then *multiHetSep* files for grey and ringed seals were created. This step is illustrated for grey seals only.

```
mkdir $work/msmc
msmc=$work/msmc
mkdir $msmc/ctgs_grey
mkdir $msmc/multi_grey

module load python-data # needed to run the $genmulti python script
```

```
SAMPLES="smp1 smp2 smp3 smp4 smp5 smp6" # list all 12 (2*6) samples here
VCF=$phasing/grey_phased.vcf.gz
FAI=$ref.fa.fai
for chr in {001..499}; do
 CHR=Hfa$chr
 len=`grep -w $CHR $FAI | cut -f2`
 if [ $len -gt 100000 ]; then
  list=""
  for SMP in $SAMPLES; do
   $bcftools view -s $SMP -r $CHR $VCF -Oz -o $msmc/ctgs_grey/$SMP.$chr.vcf.gz
   list=$list" "$msmc/ctgs_grey/$SMP.$chr.vcf.gz
  done
  $genmulti --mask $contigs/$CHR.mask.bed.gz $list \
  > $msmc/multi_grey/chr${chr}.multihetsep.txt 2> $msmc/multi_grey/log.txt
  rm $list
 fi
done
```

Demographic analysis

Then the MSMC2 analysis was run for Baltic and Atlantic grey seals and Arctic ringed seals, each analysis was done with 4 samples at a time. The first 6 samples in `$msmc/multi_grey/chr*.multihetsep.txt` files refer to the Baltic samples and the last 6 to the Atlantic samples. As this step is computationally quite intensive, the individual MSMC2 analyses were submitted as separate jobs or in groups of 2-3. The sample names are arbitrary.

This step is illustrated for grey seals, the analysis was done in the same way for rigned seals.

Baltic grey seals:

```
mkdir $msmc/output_grey

samples=(C D S E O G)
for i in $(seq 0 4); do
  for j in $(seq $(( $i + 1 )) 5); do
    for k in $(seq $(( $j + 1 )) 5); do
      for l in $(seq $(( $k + 1)) 5); do
        file=B${samples[i]}${samples[j]}${samples[k]}${samples[l]}
        $msmc2 -t 12 -p 27*1+1*2+1*3 -I $(( $i*2 )),$(( $i*2+1 )),$(( $j*2 )), \
        $(( $j*2+1 )),$(( $k*2 )),$(( $k*2+1 )),$(( $l*2 )),$(( $l*2+1 )) \
        -o $msmc/output_grey/$file $msmc/multi_grey/chr*.multihetsep.txt \
        &> $msmc/output_grey/$file.runlog
      done
    done
  done
done
```

Atlantic grey seals:

```
samples=(T Y N W I U)
for i in $(seq 6 10); do
  for j in $(seq $(( $i + 1 )) 11); do
    for k in $(seq $(( $j + 1 )) 11); do
      for l in $(seq $(( $k + 1)) 11); do
```

```
        a=$i-6
        b=$j-6
        c=$k-6
        d=$l-6
        file=A${samples[a]}${samples[b]}${samples[c]}${samples[d]}
        $msmc2 -t 12 -p 27*1+1*2+1*3 -I $(( $i*2 )),$(( $i*2+1 )),$(( $j*2 )), \
        $(( $j*2+1 )),$(( $k*2 )),$(( $k*2+1 )),$(( $l*2 )),$(( $l*2+1 )) \
        -o $msmc/output_grey/$file $msmc/multi_grey/chr*.multihetsep.txt \
        &> $msmc/output_grey/$file.runlog
      done
    done
  done
done
```

Isolation-migration inference

The MSMC-IM analysis was performed on relative Cross-Coalescence-Rate (*rCCR*) outputs of MSMC2 analysis. This was only done for grey seals. For that MSMC2 analyses were performed with 2 instead of 4 samples.

Baltic grey seals:

```
samples=(C D S E O G)
for i in $(seq 0 4); do
  for j in $(seq $(( $i + 1 )) 5); do
    file=B${samples[i]}${samples[j]}
    $msmc2 -t 12 -p 27*1+1*2+1*3 -I $(( $i*2 )),$(( $i*2+1 )),$(( $j*2 )),$(( $j*2+1 )) \
    -o $msmc/output_grey/$file $msmc/multi_grey/chr*.multihetsep.txt \
     &> $msmc/output_grey/$file.runlog
  done
done
```

Atlantic grey seals:

```
samples=(T Y N W I U)
for i in $(seq 6 10); do
  for j in $(seq $(( $i + 1 )) 11); do
    a=$i-6
    b=$j-6
    file=A${samples[a]}${samples[b]}
    $msmc2 -t 12 -p 27*1+1*2+1*3 -I $(( $i*2 )),$(( $i*2+1 )),$(( $j*2 )),$(( $j*2+1 )) \
    -o $msmc/output_grey/$file $msmc/multi_grey/chr*.multihetsep.txt \
    &> $msmc/output_grey/$file.runlog
  done
done
```

Then the *rCCR* was calculated. As this step is computationally quite intensive, the individual *rCCR* analyses were submitted as separate jobs or in groups of 2-3.

```
samples=(C D S E O G T Y N W I U)
for i in $(seq 0 4); do
  for j in $(seq 6 10); do
    file=B${samples[i]}${samples[i+1]}_A${samples[j]}${samples[j+1]}
    $msmc2 -t 12 -p 27*1+1*2+1*3 -I $(( $i*2 ))-$(( $j*2 )),$(( $i*2 ))-$(( $j*2+1 )), \
    $(( $i*2 ))-$(( $j*2+2 )),$(( $i*2 ))-$(( $j*2+3 )), \
    $(( $i*2+1 ))-$(( $j*2)),$(( $i*2+1 ))-$(( $j*2+1 )), \
    $(( $i*2+1 ))-$(( $j*2+2 )),$(( $i*2+1 ))-$(( $j*2+3 )), \
    $(( $i*2+2 ))-$(( $j*2 )),$(( $i*2+2 ))-$(( $j*2+1 )), \
    $(( $i*2+2 ))-$(( $j*2+2 )),$(( $i*2+2 ))-$(( $j*2+3 )), \
    $(( $i*2+3 ))-$(( $j*2 )),$(( $i*2+3 ))-$(( $j*2+1 )), \
    $(( $i*2+3 ))-$(( $j*2+2 )),$(( $i*2+3 ))-$(( $j*2+3 )) \
    -o $msmc/output_grey/$file $msmc/multi_grey/chr*.multihetsep.txt \
    &> $msmc/output_grey/$file.runlog
  done
done
```

Then the respective MSMC2 and *rCCR* analysis were combined, so that the samples match.

```
samples=(C D S E O G T Y N W I U)
for i in $(seq 0 4); do
  for j in $(seq 6 10); do
    b=i+1
    a=j+1
    Bfile=B${samples[i]}${samples[b]}
    Afile=A${samples[j]}${samples[a]}
    $combinecc $msmc/output_grey/${Bfile}_${Afile}.final.txt \
    $msmc/output_grey/$Bfile.final.txt msmc/output_grey/$Afile.final.txt \
    > $msmc/output_grey/c_${Bfile}_${Afile}.final.txt
  done
done
```

Based on these combined files, MSMC-IM was run.

```
samples=(C D S E O G T Y N W I U)
for i in $(seq 0 4); do
  for j in $(seq 6 10); do
    b=i+1
    a=j+1
    Bfile=B${samples[i]}${samples[b]}
    Afile=A${samples[j]}${samples[a]}
    $msmc_im -beta 1e-8,1e-6 -mu 1.826e-8 -o $msmc/output_grey/c_${Bfile}_${Afile} \
    $msmc/output_grey/c_${Bfile}_${Afile}.final.txt
  done
done
```

Convert time boundatries and coalescence rates

The MSMC2 and MSMC-IM output data was then processed in R.

Demographic analysis:

This analysis is illustrated for Atlantic grey seals only.

```
setwd("work") # set the working directory accordingly
library(dplyr) # load library

# set mutation rate and generation time
mu=1.826e-8
gt = 10
```

```
# read in the data
grey_atl <- c("ANWIU", "ATNIU", "ATNWI", "ATNWU", "ATWIU", "ATYIU", "ATYNI", \
              "ATYNU", "ATYNW", "ATYWI","ATYWU", "AYNIU", "AYNWI", "AYNWU", "AYWIU")

dat_grey_atl <- c()
for(p in grey_atl) {
  tmp <- read.table(paste0("msmc/output_grey/",p,".final.txt"),head=T)
  dat_grey_atl <- rbind.data.frame(dat_grey_atl, cbind.data.frame(pop=p,tmp))
}

# convert time boundaries
dat_grey_atl <- dat_grey_atl %>% mutate(time=left_time_boundary/mu*gt,ne=(1/lambda)/(2*mu))

# combine with other study units
dat.msmc <- rbind(dat_grey_atl, dat_grey_bal, dat_ring_arc)

# dat.msmc can then be used for plotting
```

Isolation-Migration analysis:

```
# read in the data
dat.im <- c()
for(file in dir("msmc/output_grey/","*MSMC_IM.estimates.txt")){
  tmp <- read.table(paste0("msmc/output_grey/",file),head=T)
  pair <- substr(file,1,13)
  tmp$pair <- pair
  tmp$right_time_boundary = c(tmp$left_time_boundary[2:length(tmp$left_time_boundary)],Inf)
  dat.im <- rbind.data.frame(dat.im,tmp)
}

# cut 'm' at 0.999
dat.im$m_prime <- if_else(dat.im$M <= 0.999, dat.im$m, 1e-30)

# dat.im can then be used for plotting
```

References

Olkkonen, E., & Löytynoja, A. (2023). Analysis of population structure and genetic diversity in low-variance Saimaa ringed seals using low-coverage whole-genome sequence data. STAR protocols, 4(4), 102567. https://doi.org/10.1016/j.xpro.2023.102567.