

Supplementary Material:

Nangura spinosa population genetics scripts

Nicholas Bail

2025-03-06

Contents

Intent	5
Set Up	5
Field Data and individual IDs	6
Genetic data (from vcf to genlight)	8
‘Wild’-only datasets	10
Sensible filter of full-sibs	11
Genetic distance matrices (mdist)	12
Geographic distance matrices	12
Genind format	15
Analysis Scripts	15
1. DAPC	15
Species-scale structure	16
Fine-scale structure within: Nangur	16
Fine-scale structure within: Oakview	16
2. Netviewr	17
Run netview analysis	17
Plot results	17
Supplementary Figures	18
With captive borns and 2006-2007 samples	18
With FullSib filter	18

3. Hardy-Weinberg Proportion Testing	19
spinosa_Dataset1_Nan	19
Identifying global deviations from HWP	19
Assessing nature of global deviations from HWP	20
Identity of outlier loci	22
spinosa_Dataset1_Oak	22
Identifying global deviations from HWP	22
Assessing nature of global deviations from HWP	23
Identity of outlier loci	24
spinosa_Dataset1_Oak_wild	25
Identifying global deviations from HWP	25
Assessing nature of global deviations from HWP	25
Identity of outlier loci	27
spinosa_Dataset1_Nan_FSfilter	28
Identifying global deviations from HWP	28
Assessing nature of global deviations from HWP	28
Identity of outlier loci	29
spinosa_Dataset1_Oak_wild_FSfilter	29
Identifying global deviations from HWP	29
Assessing nature of global deviations from HWP	30
Identity of outlier loci	31
4. Diversity and Inbreeding	31
Population-wide genomic diversity	31
Comparing MLH through time	32
Checking that heterozygosity is not associated with poorer read depth	34
Table of population-level diversity indices	35
spinosa_Dataset2	35
Genetic Diversity Table for paper	38
FSfilter	41
Genetic Diversity Table for paper	45
Genetic Capture	48
Oakview	48
Nangur	49
Both (private and fixed alleles)	50
Combine the dataframes	50
Short-cut: use saved data	51

Plot the results	51
Supplementary: Effect of sample size	51
Diversity	53
private and fixed alleles	54
5. AMOVA and Fst	55
AMOVA	55
Species Level	55
spinosa_Dataset1	56
spinosa_Dataset1_FSfilter	57
Oakview K=2	58
spinosa_Dataset1_Oak	58
spinosa_Dataset1_Oak_FSfilter	59
Oakview K=3	60
spinosa_Dataset1_Oak_wild	60
spinosa_Dataset1_Oak_wild_FSfilter	61
Nangur K=2 (v1. East/West)	62
spinosa_Dataset1_Nan	62
spinosa_Dataset1_Nan_FSfilter	63
Nangur K=2 (v2. Far East/West)	65
spinosa_Dataset1_Nan	65
Nangur K=3	66
spinosa_Dataset1_Nan	66
Private alleles	67
Species Level	67
Oakview	67
Nangur	67
Fst	68
6. Genomic Distance	68
Prepare dataframes	69
Individual mean pairwise genomic distance	70
spinosa_Dataset1	70
Does geographic distance explain genomic distance?	71
spinosa_Dataset1_Nan	71
spinosa_Dataset1_Oak	72
Combine the plots	73

Average genomic distance between and within populations	74
Without full-sib filter	74
With full-sib filter	75
Average pairwise genomic distance between full sibs and halvesibs.	76
Fullsibs	76
Nangur	77
Oakview	77
Half-sibs	78
Nangur	79
Oakview	80
The genomic distance plot with full and halvesibs marked	80
7. Neighbourhood Radius Size	81
Infering Neighbourhood radius	82
genind_spinosa_Dataset1_Nan	82
genind_spinosa_Dataset1_Oak_wild	85
genind_spinosa_Dataset2_wild	89
genind_spinosa_Dataset1_Nan_FSfilter	92
genind_spinosa_Dataset1_Oak_wild_FSfilter	95
8. Effective Population Size	99
File path for Ne estimator	100
Estimate Ne for full populations	100
spinosa_Dataset1_Nan	100
spinosa_Dataset1_Nan_FSfilter	101
spinosa_Dataset1_Oak	103
spinosa_Dataset1_Oak_wild_FSfilter	104
Plotting	106
Data entry	106
Short-cut: use saved data	109
ggplot	109
8b. Effective Population Size (effect of sample size)	113
File path for Ne estimator	113
spinosa_Dataset1_Oak	113
spinosa_Dataset1_Nan	115
Short-cut: use saved data	116
Combined plot	116

9. Mapping	117
‘NetviewR’ map	117
MLH (inbreeding) map	119
Related individuals map	120
Redefine families	120
Fullsibs	121
Halfsibs	123
Reference Map	123
References	126
Packages	126
Markdown Appendix 1: How files for Netview were generated	135
Markdown Appendix 2: How genotypes were converted to Colony format	136

Intent

This R Markdown is intended to be a repository of scripts used in the analysis of SNP data for *Nangura spinosa*. It begins with three filtered datasets exported from vcftools, and covers the final version of each of the population genetic analyses that followed. Combined with the provided data, it should make all analyses completely reproducible.

For any additional information, please contact the corresponding author.

Set Up

This first ‘set up’ section includes scripts that load all of the packages required, and reads in almost all of the dataframes required for the actual analyses scripts.

```
#### Directory ####
# This R Markdown works best if you set the working directory to this file's
# location.

# This code has only been fully tested in R version 4.3.3

#### Packages to load ###
# Please install all of the following packages, before loading.
library(devtools) # This needs to be the first package installed.
library(tidyverse)
library(knitr)
library(RColorBrewer)
library(vcfR)
library(adegenet)
library(igraph)
#library(ggtern)
```

```

library(reshape2)
library(geodist)
library(ggbridges)
library(viridis)
library(poppr)
library(patchwork)
library(ggbreak)
library(HardyWeinberg)
library(ozmaps)
library(sf)
library(splitstackshape)
library(leaflet)
library(netviewr) # Requires github installation, see: https://github.com/esteinig/netviewr
library(dartR) # This is now a legacy version, see: https://github.com/green-striped-gecko/dartR/wiki/Installation
# Package: TinyTex is used to generate R Markdown pdfs
library(sGD) # Requires github installation: install_github("Andrew-Shirk/sGD")

```

Field Data and individual IDs

Load in and modify field data relevant to all analyses.

```

# Read in the field data for this species provided by Dan Ferguson
NanguraSpinosa.dat <- read.csv(file = '../data/spinosa_sample_data.csv')

# That file does not include some individuals that were collected in an older project.
# It also needs to be in the correct order for some of the genetic results.
# Hence, I import the following file,
# which was produced by Plink while generating distance matrices from the genetic data.

# Read in the id file
NanguraSpinosa.id <- read.delim(file = "../data/spinosa_sample_id.txt",
                                sep = "\t", header = F)

# For total number of samples (not individuals) sent to AGRF (pre-filtering)
c(NanguraSpinosa.dat$Field.No. [
  NanguraSpinosa.dat$Field.No. %in% NanguraSpinosa.id$V1,
  'BSU0454', 'Oakview01', 'Oakview02', 'Oakview03', 'Oakview04', 'Oakview05',
  'Nangur01', 'Nangur02', 'Nangur03', 'Nangur04', 'Nangur05') |>
  # These samples are not in the list because they are from the
  # 2006-2007 samples (added to dataframe later) or were excluded early due to
  # poor quality (BSU0454, Oakview05, & Nangur05).
  length() + 3
# 3 samples sent to AGRF outside of the 96 well (95 samples + blank) plate.
# They were for establishment.
# Note BSU0454 is a wildborn captive individual

# Merge the ids from the plink file with the field data (as the list of
# individuals in the field data does not exactly match the ones in the analysis
# due to individuals being excluded/included)
NanguraSpinosa.dat <- merge(x = NanguraSpinosa.id, y = NanguraSpinosa.dat,
                             by.x = 'V1', by.y = 'Field.No.', all.x = TRUE) %>%

```

```

arrange(factor(V1, levels = NanguraSpinosa.id$V1))
# This line of code makes sure that the individuals are ordered in the same way
# as the distance matrix produced by Plink
# and hence stays consistent with other datasets.

# Rename the column holding individual's names.
NanguraSpinosa.dat <- NanguraSpinosa.dat |>
  rename('INDV' = V1)

# Add a population name for the individuals that were not included in the field
# data.

NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Nangur01'] <- 'Nangur'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Nangur02'] <- 'Nangur'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Nangur03'] <- 'Nangur'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Nangur04'] <- 'Nangur'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Oakview01'] <-
  'Oakview'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Oakview02'] <-
  'Oakview'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Oakview03'] <-
  'Oakview'
NanguraSpinosa.dat$Population[NanguraSpinosa.dat$INDV == 'Oakview04'] <-
  'Oakview'

# Create a new column, year, which identifies them by the sampling period
# in which they were collected
NanguraSpinosa.dat$Year <- '2020-2021'

NanguraSpinosa.dat$Year[NanguraSpinosa.dat$INDV == 'Oakview01' |
  NanguraSpinosa.dat$INDV == 'Oakview02' |
  NanguraSpinosa.dat$INDV == 'Oakview03' |
  NanguraSpinosa.dat$INDV == 'Oakview04' |
  NanguraSpinosa.dat$INDV == 'Nangur01' |
  NanguraSpinosa.dat$INDV == 'Nangur02' |
  NanguraSpinosa.dat$INDV == 'Nangur03' |
  NanguraSpinosa.dat$INDV == 'Nangur04'] <-
  '2006-2007'

# Order the Year category

NanguraSpinosa.dat$Year <- factor(NanguraSpinosa.dat$Year,
  levels = c('2006-2007', '2020-2021'))

```

In order to aid visualisation of Netview plots, arbitrary colours are assigned to geographic clusters of samples. This is also placed on a map so that the corresponding colours can be shared. The column in the .csv is called `sample_cluster` and also highlights if individuals were born in captivity (but it doesn't highlight if they are now in captivity). Here I create these and add it to the `NanguraSpinosa.dat` metadata document.

```

# cluster_colours based on sample_clusters from West to East
NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
  'Nangur_farthest'] <- 'olivedrab1'

```

```

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Nangur_midwest'] <- 'green'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Nangur_mideast'] <- 'forestgreen'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Nangur_fareast'] <- 'olivedrab'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Oakview_farwest'] <- 'darkslategray1'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Oakview_north'] <- 'mediumpurple1'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Oakview_midwest'] <- 'deepskyblue'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Oakview_south'] <- 'mediumorchid1'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Oakview_mideast'] <- 'royalblue'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'Oakview_fareast'] <- 'navy'

NanguraSpinosa.dat$cluster_colour[NanguraSpinosa.dat$Sample_cluster ==
    'captive'] <- 'yellow'

NanguraSpinosa.dat$cluster_outline <- 'black'

NanguraSpinosa.dat$cluster_outline[
    NanguraSpinosa.dat$Sample_cluster != 'captive' &
    NanguraSpinosa.dat$captive_wild == 'captive'] <- 'darkorange'

NanguraSpinosa.dat$cluster_outlinewidth[
    NanguraSpinosa.dat$Sample_cluster != 'captive' &
    NanguraSpinosa.dat$captive_wild != 'captive'] <- 0.5

NanguraSpinosa.dat$cluster_outlinewidth[
    NanguraSpinosa.dat$Sample_cluster != 'captive' &
    NanguraSpinosa.dat$captive_wild == 'captive'] <- 2

```

Genetic data (from vcf to genlight)

Load in genetic datasets from vcf format having been exported from vcftools version 0.1.16 (Danecek et al., 2011), and Stacks version 2.62 (Catchen et al., 2011; Catchen et al., 2013). Dataset1, 2, and 3, and the methods by which they are derived, are described in the paper.

```

# Read in vcf files for all relevant genetic datasets.
#
vcfR_spinosa_Dataset1_preclean <- read.vcfR(
  file = "../data/vcf_files/spinosa_Dataset1_preclean.recode.vcf")

vcfR_spinosa_Dataset1_Nan <- read.vcfR(
  file = "../data/vcf_files/spinosa_Dataset1_Nan.recode.vcf")

vcfR_spinosa_Dataset1_Oak <- read.vcfR(
  file = "../data/vcf_files/spinosa_Dataset1_Oak.recode.vcf")

vcfR_spinosa_Dataset2 <- read.vcfR(
  file = "../data/vcf_files/spinosa_Dataset2.recode.vcf")

vcfR_spinosa_Dataset3 <- read.vcfR(
  file = "../data/vcf_files/spinosa_Dataset3.recode.vcf")

```

Convert to genlight and delete the vcfR file

```

# Convert to genlight
genlight_spinosa_Dataset1_preclean <- vcfR2genlight(
  vcfR_spinosa_Dataset1_preclean)

genlight_spinosa_Dataset1_Nan <- vcfR2genlight(vcfR_spinosa_Dataset1_Nan)

genlight_spinosa_Dataset1_Oak <- vcfR2genlight(vcfR_spinosa_Dataset1_Oak)

genlight_spinosa_Dataset2 <- vcfR2genlight(vcfR_spinosa_Dataset2)

genlight_spinosa_Dataset3 <- vcfR2genlight(vcfR_spinosa_Dataset3)

# Essential compliance check and correction for genlight files

genlight_spinosa_Dataset1_preclean <- gl.compliance.check(
  genlight_spinosa_Dataset1_preclean)

genlight_spinosa_Dataset1_Nan <- gl.compliance.check(
  genlight_spinosa_Dataset1_Nan)

genlight_spinosa_Dataset1_Oak <- gl.compliance.check(
  genlight_spinosa_Dataset1_Oak)

genlight_spinosa_Dataset2 <- gl.compliance.check(
  genlight_spinosa_Dataset2)

genlight_spinosa_Dataset3 <- gl.compliance.check(
  genlight_spinosa_Dataset3)

# Remove original vcfR file to save memory:

```

```

rm(vcfR_spinosa_Dataset1_preclean)

rm(vcfR_spinosa_Dataset1_Nan)

rm(vcfR_spinosa_Dataset1_Oak)

rm(vcfR_spinosa_Dataset2)

rm(vcfR_spinosa_Dataset3)

# Create a full Dataset1 genlight
# (as it is not yet represented as a single genlight)

genlight_spinosa_Dataset1 <- gl.keep.ind(
  genlight_spinosa_Dataset1_preclean,
  ind.list = genlight_spinosa_Dataset1_preclean$ind.names[
    genlight_spinosa_Dataset1_preclean$ind.names %in% c(
      genlight_spinosa_Dataset1_Oak$ind.names,
      genlight_spinosa_Dataset1_Nan$ind.names)])

# Perform compliance check on Dataset 1
genlight_spinosa_Dataset1 <- gl.compliance.check(
  genlight_spinosa_Dataset1)

```

‘Wild’-only datasets

For some analyses, the presence of captive-born individuals will be confusing. This is because it includes offspring of parents from potentially disparate gene pools and hence does not allow me to represent the patterns present in the wild. Hence, I produce some datasets that have only ‘wild-born’ individuals. This only applies to the Oakview datasets.

```

# Drop individuals marked as being from captivity
genlight_spinosa_Dataset1_preclean_wild <- gl.drop.ind(
  genlight_spinosa_Dataset1_preclean,
  ind.list = genlight_spinosa_Dataset1_preclean$ind.names[
    genlight_spinosa_Dataset1_preclean$ind.names %in%
    NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'captivity']])

genlight_spinosa_Dataset1_wild <- gl.keep.ind(
  genlight_spinosa_Dataset1_preclean_wild,
  ind.list = genlight_spinosa_Dataset1_preclean_wild$ind.names[
    genlight_spinosa_Dataset1_preclean_wild$ind.names %in% c(
      genlight_spinosa_Dataset1_Oak$ind.names,
      genlight_spinosa_Dataset1_Nan$ind.names)])

genlight_spinosa_Dataset1_Oak_wild <- gl.drop.ind(
  genlight_spinosa_Dataset1_Oak,
  ind.list = genlight_spinosa_Dataset1_Oak$ind.names[
    genlight_spinosa_Dataset1_Oak$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'captivity']])

```

```
genlight_spinosa_Dataset2_wild <- gl.drop.ind(
  genlight_spinosa_Dataset2, ind.list = genlight_spinosa_Dataset2$ind.names[
    genlight_spinosa_Dataset2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'captivity']] )
```

Sensible filter of full-sibs

Here I perform a ‘sensible’ filter of full-sibs. All results will be analysed with and without this filter. Previously, I attempted to use methods recommended in papers as robust (e.g., Yank-2 from Waples & Anderson, 2017). However, I realised that in doing that I ignored their main advice which was to consider the context of my data and its affect on filtering. For example, Yank-2 doesn’t make sense if I have much smaller groups of full-sibs, many of which are parents and offspring. Instead, I use the following rules:

- In pairs of full-sibs I remove one, preferably an offspring if it is a parent-offspring pair (very traditional).
- In groups that clearly have two adults + offspring, I remove the offspring and leave the unrelated adults.
- Similarly, if I have a trio with one individual that is a full sib with the other two, but the other two are not full sibs, I take the connecting individual.
- In a trio of mutual full-sibs, I remove all but one.

This also has the advantage of making the data more homogeneous with respect to generation.

Remove individuals from each fulls sib family, except for one. List them here.

```
genlight_spinosa_Dataset1_Oak_FSfilter <- gl.drop.ind(
  genlight_spinosa_Dataset1_Oak, ind.list = c(
    'BSU0548', 'BSU0469', 'BSU0507', 'BSU0543', '
    BSU0544', 'BSU0550', 'BSU0432', 'BSU0499',
    'BSU0500', 'BSU0444', 'BSU0504', 'BSU0430'))

genlight_spinosa_Dataset1_Oak_wild_FSfilter <- gl.drop.ind(
  genlight_spinosa_Dataset1_Oak_wild, ind.list = c(
    'BSU0548', 'BSU0469', 'BSU0507', 'BSU0543',
    'BSU0544', 'BSU0550', 'BSU0432', 'BSU0499',
    'BSU0500', 'BSU0444', 'BSU0504', 'BSU0430'))

genlight_spinosa_Dataset1_Nan_FSfilter <- gl.drop.ind(
  genlight_spinosa_Dataset1_Nan, ind.list = c(
    'BSU0571', 'BSU0541', 'BSU0533', 'BSU0530',
    'BSU0574', 'BSU0523', 'BSU0525', 'BSU0524',
    'BSU0522', 'BSU0527'))

genlight_spinosa_Dataset1_wild_FSfilter <- gl.drop.ind(
  genlight_spinosa_Dataset1_wild, ind.list = c(
    'BSU0548', 'BSU0469', 'BSU0507', 'BSU0543',
    'BSU0544', 'BSU0550', 'BSU0432', 'BSU0499',
    'BSU0500', 'BSU0444', 'BSU0504', 'BSU0430',
    'BSU0571', 'BSU0541', 'BSU0533', 'BSU0530',
    'BSU0574', 'BSU0523', 'BSU0525', 'BSU0524',
    'BSU0522', 'BSU0527'))

genlight_spinosa_Dataset2_FSfilter <- gl.drop.ind(
  genlight_spinosa_Dataset2, ind.list = c(
```

```

'BSU0548', 'BSU0469', 'BSU0507', 'BSU0543',
'BSU0544', 'BSU0550', 'BSU0432', 'BSU0499',
'BSU0500', 'BSU0444', 'BSU0504', 'BSU0430',
'BSU0571', 'BSU0541', 'BSU0533', 'BSU0530',
'BSU0574', 'BSU0523', 'BSU0525', 'BSU0524',
'BSU0522', 'BSU0527'))

genlight_spinosa_Dataset2_wild_FSfilter <- gl.drop.ind(
  genlight_spinosa_Dataset2_wild, ind.list = c(
    'BSU0548', 'BSU0469', 'BSU0507', 'BSU0543',
    'BSU0544', 'BSU0550', 'BSU0432', 'BSU0499',
    'BSU0500', 'BSU0444', 'BSU0504', 'BSU0430',
    'BSU0571', 'BSU0541', 'BSU0533', 'BSU0530',
    'BSU0574', 'BSU0523', 'BSU0525', 'BSU0524',
    'BSU0522', 'BSU0527'))

```

Genetic distance matrices (mdist)

These are used for netviewr analyses. See the appendix for more details on how these files were produced.

```

mdist_spinosa_Dataset1_preclean <- read.delim(
  file = '../data/other_files/dist_spinosa_Dataset1_preclean.mdist',
  sep = "\t", header = F) %>%
  data.matrix()

```

Generate mdist data using spinosa_Dataset1_preclean and then lists of individuals from other data types.

```

#cleanPop_wild dataset is created by keeping individuals that are in
# cleanNan and cleanOak only, but not captive born individuals
mdist_spinosa_Dataset1_wild <- mdist_spinosa_Dataset1_preclean[
  NanguraSpinosa.id$V1 %in% genlight_spinosa_Dataset1_wild$ind.names,
  NanguraSpinosa.id$V1 %in% genlight_spinosa_Dataset1_wild$ind.names]

# FSfilter
mdist_spinosa_Dataset1_wild_FSfilter <- mdist_spinosa_Dataset1_preclean[
  NanguraSpinosa.id$V1 %in% genlight_spinosa_Dataset1_wild_FSfilter$ind.names,
  NanguraSpinosa.id$V1 %in% genlight_spinosa_Dataset1_wild_FSfilter$ind.names]

```

Geographic distance matrices

These geographic distance matrices are used by sGD and other analyses that need geographic distances. One is made for each subset of individuals for convenience. I use geodesic distances and the package geodist. Remember that captive born individuals don't have lat longs.

```

# Extract the lat longs ONLY for individuals that are present in the
# genlight file and combine into a single file with latitudes and longitudes.
# Make sure they are correctly aligned.

latlong_spinosa <- subset(
  NanguraSpinosa.dat,

```

```

subset = INDV %in% genlight_spinosa_Dataset1_preclean_wild$ind.names,
select = c(INDV, Latitude, Longitude))

latlong_spinosa_Dataset1 <- subset(
  NanguraSpinoso.dat,
  subset = INDV %in% genlight_spinosa_Dataset2_wild$ind.names,
  select = c(INDV, Latitude, Longitude))

latlong_spinosa_Dataset1_Nan <- subset(
  NanguraSpinoso.dat,
  subset = INDV %in% genlight_spinosa_Dataset1_Nan$ind.names,
  select = c(INDV, Latitude, Longitude))

latlong_spinosa_Dataset1_Oak <- subset(
  NanguraSpinoso.dat,
  subset = INDV %in% genlight_spinosa_Dataset1_Oak_wild$ind.names,
  select = c(INDV, Latitude, Longitude))

latlong_spinosa_Dataset1_Nan_FSfilter <- subset(
  NanguraSpinoso.dat,
  subset = INDV %in% genlight_spinosa_Dataset1_Nan_FSfilter$ind.names,
  select = c(INDV, Latitude, Longitude))

latlong_spinosa_Dataset1_Oak_FSfilter <- subset(
  NanguraSpinoso.dat,
  subset = INDV %in% genlight_spinosa_Dataset1_Oak_wild_FSfilter$ind.names,
  select = c(INDV, Latitude, Longitude))

# Order the latlongs data so it is the same as the genind data.
# Do this by first adding a factor based on the names in the genind object.
latlong_spinoso$INDV <- factor(
  latlong_spinoso$INDV,
  levels = unique(
    as.character(genlight_spinoso_Dataset1_preclean_wild$ind.names)))

latlong_spinoso_Dataset1$INDV <- factor(
  latlong_spinoso_Dataset1$INDV,
  levels = unique(as.character(genlight_spinoso_Dataset2_wild$ind.names)))

latlong_spinoso_Dataset1_Nan$INDV <- factor(
  latlong_spinoso_Dataset1_Nan$INDV,
  levels = unique(
    as.character(genlight_spinoso_Dataset1_Nan$ind.names)))

latlong_spinoso_Dataset1_Oak$INDV <- factor(
  latlong_spinoso_Dataset1_Oak$INDV,
  levels = unique(as.character(genlight_spinoso_Dataset1_Oak_wild$ind.names)))

latlong_spinoso_Dataset1_Nan_FSfilter$INDV <- factor(
  latlong_spinoso_Dataset1_Nan_FSfilter$INDV,
  levels = unique(
    as.character(genlight_spinoso_Dataset1_Nan_FSfilter$ind.names)))

```

```

latlong_spinosa_Dataset1_Oak_FSfilter$INDV <- factor(
  latlong_spinosa_Dataset1_Oak_FSfilter$INDV,
  levels = unique(
    as.character(genlight_spinosa_Dataset1_Oak_wild_FSfilter$ind.names)))

# Assign the new order
latlong_spinosa <- latlong_spinosa[order(latlong_spinosa$INDV), ]

latlong_spinosa_Dataset1 <- latlong_spinosa_Dataset1[
  order(latlong_spinosa_Dataset1$INDV), ]

latlong_spinosa_Dataset1_Nan <- latlong_spinosa_Dataset1_Nan[
  order(latlong_spinosa_Dataset1_Nan$INDV), ]

latlong_spinosa_Dataset1_Oak <- latlong_spinosa_Dataset1_Oak[
  order(latlong_spinosa_Dataset1_Oak$INDV), ]

latlong_spinosa_Dataset1_Nan_FSfilter <- latlong_spinosa_Dataset1_Nan[
  order(latlong_spinosa_Dataset1_Nan_FSfilter$INDV), ]

latlong_spinosa_Dataset1_Oak_FSfilter <- latlong_spinosa_Dataset1_Oak[
  order(latlong_spinosa_Dataset1_Oak_FSfilter$INDV), ]

## Calculate pairwise geodesic distances using the package geodist

distmat_spinosa <- latlong_spinosa |>
  geodist(measure = 'geodesic') |>
  as.matrix()

distmat_spinosa_Dataset1 <- latlong_spinosa_Dataset1 |>
  geodist(measure = 'geodesic') |>
  as.matrix()

distmat_spinosa_Dataset1_Nan <- latlong_spinosa_Dataset1_Nan |>
  geodist(measure = 'geodesic') |>
  as.matrix()

distmat_spinosa_Dataset1_Oak <- latlong_spinosa_Dataset1_Oak |>
  geodist(measure = 'geodesic') |>
  as.matrix()

distmat_spinosa_Dataset1_Nan_FSfilter <-
  latlong_spinosa_Dataset1_Nan_FSfilter |>
  geodist(measure = 'geodesic') |>
  as.matrix()

distmat_spinosa_Dataset1_Oak_FSfilter <-
  latlong_spinosa_Dataset1_Oak_FSfilter |>
  geodist(measure = 'geodesic') |>
  as.matrix()

```

Genind format

Some software (such as sGD) require a genind format. Here I provide those conversions. Not all of these datasets will necessarily be used in the analysis.

```
# Create genind from genlight using dartR.

genind_spinosa_Dataset1_preclean <- gl2gi(genlight_spinosa_Dataset1_preclean)

genind_spinosa_Dataset1_preclean_wild <- gl2gi(
  genlight_spinosa_Dataset1_preclean_wild)

genind_spinosa_Dataset1_Nan <- gl2gi(genlight_spinosa_Dataset1_Nan)

genind_spinosa_Dataset1_Oak <- gl2gi(genlight_spinosa_Dataset1_Oak)

genind_spinosa_Dataset1_Oak_wild <- gl2gi(genlight_spinosa_Dataset1_Oak_wild)

genind_spinosa_Dataset2 <- gl2gi(genlight_spinosa_Dataset2)

genind_spinosa_Dataset2_wild <- gl2gi(genlight_spinosa_Dataset2_wild)

# Remove individuals according to the full-sib filter...
genind_spinosa_Dataset1_Oak_FSfilter <- gl2gi(
  genlight_spinosa_Dataset1_Oak_FSfilter)

genind_spinosa_Dataset1_Oak_wild_FSfilter <- gl2gi(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter)

genind_spinosa_Dataset1_Nan_FSfilter <- gl2gi(
  genlight_spinosa_Dataset1_Nan_FSfilter )

genind_spinosa_Dataset1_wild_FSfilter <- gl2gi(
  genlight_spinosa_Dataset1_wild_FSfilter )

genind_spinosa_Dataset2_FSfilter <- gl2gi(genlight_spinosa_Dataset2_FSfilter )

genind_spinosa_Dataset2_wild_FSfilter <- gl2gi(
  genlight_spinosa_Dataset2_wild_FSfilter )
```

Data must be converted from genlight to genind for analysis in the software sGD.

Analysis Scripts

Here, scripts for each of several major analyses in the study are included.

1. DAPC

Connectivity and Structure: DAPC:

The purpose of this analysis is:

- to use K-means clustering and the BIC to determine what clustering of individuals best fits the data (the K (# of clusters) with lowest BIC). This is aimed at scale of the species but also more fine-scale structuring within the two positions.
- To use K-means clustering to identify how individuals can be grouped at each K.
- To use the DAPC to summarise how genetic variation is structured between putative groupings.

Species-scale structure

Determine the number of clusters within Nangura spinosa dataset using k-means clustering and the BIC. The optimal K is preceded by a drop in the BIC representing a sudden decrease in the amount of unexplained deviance.

All species level datasets yield the same result.

I have set this up so each BIC plot can be investigated separately before making decisions about the number of clusters, although that can be decided a priori.

```
# Run k-means clustering on DAPC.
# This chunk is designed to fail, and hence produce a BIC plot.
k_means_spinosa_Dataset1 <- find.clusters(
  genlight_spinosa_Dataset1, max.n.clust = 30, n.pca = 100)

# To actually determine the membership of clusters:
#k_means_spinosa_Dataset1 <- find.clusters(genlight_spinosa_Dataset1,
# max.n.clust = 30, n.pca = 100, n.clust = 2)

# Repeat the k-means clustering but only with wild individuals.
k_means_spinosa_Dataset1_wild <- find.clusters(
  genlight_spinosa_Dataset1_wild, max.n.clust = 30, n.pca = 50)
```

Fine-scale structure within: Nangur

```
# Run k-means clustering on DAPC.

# This chunk is designed to fail, and hence produce a BIC plot
find.clusters(genlight_spinosa_Dataset1_Nan, max.n.clust = 22, n.pca = 22)

find.clusters(genlight_spinosa_Dataset1_Nan_FSfilter, n.pca = 22)
```

Despite K = 1 being the optimal clustering, I consider alternative clustering options and plot them.

Fine-scale structure within: Oakview

Note that I exclude captive-born individuals for this analysis.

```
# Run k-means clustering on DAPC.

# This chunk is designed to fail, and hence produce a BIC plot
find.clusters(genlight_spinosa_Dataset1_Oak_wild,
  max.n.clust = 40, n.pca = 100)
```

```
find.clusters(genlight_spinosa_Dataset1_Oak_wild_FSfilter,
              max.n.clust = 30, n.pca = 100)
```

Despite $K = 1$ being the optimal clustering, I consider alternative clustering options and plot them.

2. Netviewr

Connectivity and Structure: NetviewR:

The purpose of this analysis is:

- To use a network style analysis (netviewr) to map genomic distances to individuals in the dataset.
- To use a method with few assumptions to observe natural patterns in how individuals cluster within the populations.
- To see how closely subclusters link together.
- To see population structure at various spatial scales.

Run netview analysis

```
# Generate a graph for k-nn = 1:30
# Use the dataset that only has wild individuals from the 'cleanPop'

netview_spinosa_Dataset1_wild <- netview(dist = mdist_spinosa_Dataset1_wild,
                                         k = 1:60)
```

Plot results

```
y = 3

#### figure_netview ####
par(mfrow = c(1, 2))

#### k-NN = 4 ####
netview_spinosa_Dataset1_wild$k7 %%%
#node_label(data = genlight_spinosa_Dataset1_wild$ind.names) %%%
  node_size(data = y, min=10, max=10) %>%
  plot_netview(vertex.color = NanguraSpinosa.dat$cluster_colour[
    NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_wild$ind.names],
               vertex.frame.color = NanguraSpinosa.dat$cluster_outline[
    NanguraSpinosa.dat$INDV %in%
    genlight_spinosa_Dataset1_wild$ind.names],
               vertex.frame.width = NanguraSpinosa.dat$cluster_outlinewidth[
    NanguraSpinosa.dat$INDV %in%
    genlight_spinosa_Dataset1_wild$ind.names],
               main = 'k-NN = 7')
```

```
#### k-NN = 21 ####
netview_spinoso_Dataset1_wild$k21 %@@
#node_label(data = genlight_spinoso_Dataset1_wild$ind.names) %@@
  node_size(data = y, min=10, max=10) %>%
plot_netview(vertex.color = NanguraSpinoso.dat$cluster_colour[
  NanguraSpinoso.dat$INDV %in% genlight_spinoso_Dataset1_wild$ind.names],
  vertex.frame.color = NanguraSpinoso.dat$cluster_outline[
    NanguraSpinoso.dat$INDV %in%
    genlight_spinoso_Dataset1_wild$ind.names],
  vertex.frame.width = NanguraSpinoso.dat$cluster_outlinewidth[
    NanguraSpinoso.dat$INDV %in%
    genlight_spinoso_Dataset1_wild$ind.names],
  main = 'k-NN = 21')
```

Supplementary Figures

With captive borns and 2006-2007 samples

```
netview_spinoso_Dataset1_preclean <- netview(
  dist = mdist_spinoso_Dataset1_preclean, k = 1:30)

netview_spinoso_Dataset1_preclean$k10 %@@
#node_label(data = genlight_spinoso_Dataset1_preclean$ind.names) %@@
  node_size(data = y, min=10, max=10) %>%
plot_netview(vertex.color = NanguraSpinoso.dat$cluster_colour[
  NanguraSpinoso.dat$INDV %in% genlight_spinoso_Dataset1_preclean$ind.names],
  vertex.frame.color = NanguraSpinoso.dat$cluster_outline[
    NanguraSpinoso.dat$INDV %in%
    genlight_spinoso_Dataset1_preclean$ind.names],
  #vertex.frame.width = NanguraSpinoso.dat$cluster_outlinewidth[
    #NanguraSpinoso.dat$INDV %in% genlight_spinoso_Dataset1_preclean$ind.names],
  main = 'k-NN = 10')
```

With FullSib filter

```
netview_spinoso_Dataset1_wild_FS <- netview(
  dist = mdist_spinoso_Dataset1_wild_FSfilter, k = 1:30)

par(mfrow = c(1, 2))

netview_spinoso_Dataset1_wild_FS$k4 %@@
#node_label(data = genlight_spinoso_Dataset1_preclean$ind.names) %@@
  node_size(data = y, min=10, max=10) %>%
plot_netview(vertex.color = NanguraSpinoso.dat$cluster_colour[
  NanguraSpinoso.dat$INDV %in%
  genlight_spinoso_Dataset1_wild_FSfilter$ind.names],
  vertex.frame.color = NanguraSpinoso.dat$cluster_outline[
    NanguraSpinoso.dat$INDV %in%
    genlight_spinoso_Dataset1_wild_FSfilter$ind.names],
```

```

        #vertex.frame.width = NanguraSpinosa.dat$cluster_outlinewidth[
        #NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_preclean$ind.names],
        main = 'k-NN = 4')

netview_spinosa_Dataset1_wild_FS$k21 %%
#node_label(data = genlight_spinosa_Dataset1_preclean$ind.names) %%
node_size(data = y, min=10, max=10) %>%
plot_netview(vertex.color = NanguraSpinosa.dat$cluster_colour[
NanguraSpinosa.dat$INDV %in%
genlight_spinosa_Dataset1_wild_FSfilter$ind.names],
vertex.frame.color =
NanguraSpinosa.dat$cluster_outline[
NanguraSpinosa.dat$INDV %in%
genlight_spinosa_Dataset1_wild_FSfilter$ind.names],
#vertex.frame.width = NanguraSpinosa.dat$cluster_outlinewidth[
#NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_preclean$ind.names],
main = 'k-NN = 21')

```

3. Hardy-Weinberg Proportion Testing

Quality Control: Hardy Weinberg Proportions (HWP):

The purpose of this analysis is to:

- Assess whether the populations of this dataset deviate significantly from expected Hardy Weinberg Proportions.
- Assess the nature of deviations from Hardy Weinberg Proportions, and guess at what the biological explanation for the deviations might be.
- To provide evidence towards whether datasets consisting of the entirety of Nangur or Oakview show a Wahlund effect (and therefore need to have internal structure).
- To identify outlier loci that are significantly out of HWP.
- To determine how to filter outlier loci.

This analysis broadly follows the approach described by Waples et al., (2015).

spinosa_Dataset1_Nan

Identifying global deviations from HWP

Here I perform a test on all loci to see if the dataset deviates from HWP proportions. I look at Nangur and Oakview separately, because it is a given that they are separate populations and not to be assessed as one.

```

# Perform a test for deviations from HWP
HWreport_spinosa_Dataset1_Nan <- gl.report.hwe(genlight_spinosa_Dataset1_Nan,
        subset = 'all', sig_only = F,
        multi_comp = T, alpha_val = 0.05,
        multi_comp_method = 'fdr')

# Count the number of significant deviations
HWreport_spinosa_Dataset1_Nan %>%
count(Sig)

```

This is an example of multiple testing. I now need to determine if there are significantly more significant deviations than expected. My critical value is 0.05. I have performed 2968 tests (that is the number of polymorphic loci in this dataset, and they are two-tailed tests. Hence the expected number of type 1 errors (false discovery rate) is: $2968 \times 0.05 = 148.45$.

What follows is a binomial hypothesis test, followed by a visual assessment of the same results, in which p values should be roughly uniformly distributed with no excess of p values below 0.05.

```
# Test if there are significantly more significant deviations than expected
pbinom(q = 203, size = 203+2765, prob = 0.05, lower.tail = FALSE)

# Assess visually.
hist(HWreport_spinosa_Dataset1_Nan$Prob, breaks = 20)
abline(h = 0.05*(203+2765), col = 'red')
```

Assessing nature of global deviations from HWP

One way to investigate further is to assess the Fis distribution and see if it is translated in a positively or negative direction. Negative indicates heterozygote excess (e.g., due to a small N_e), positive indicates heterozygote deficiency (e.g., due to a Wahlund effect). See Waples 2015 for biological processes that can cause these two issues.

```
# Calculate basic stats (particularly Fis) for each locus
locistats_spinosa_Dataset1_Nan <- gl.basic.stats(genlight_spinosa_Dataset1_Nan)

# View distribution and summary

# Create dataframe, with both the Fis values and a null distribution
# (assuming normal with mean of 0 and SD of 0.2).
Fis_comparison <- tibble(
  Obs = na.omit(locistats_spinosa_Dataset1_Nan$perloc$Fis),
  Exp = rnorm(n = length(na.omit(locistats_spinosa_Dataset1_Nan$perloc$Fis)),
    mean = 0, sd = 0.2))

# Plot a histogram showing whether Fis is translated towards positive or
# negative values.
Fis_comparison |>
  ggplot(aes(x = x) ) +
  #Top
  geom_density(aes(x = Obs, y = ..density..), fill = "#69b3a2") +
  # Bottom
  geom_density(aes(x = Exp, y = -..density..), fill = "#404080") +
  theme_bw() +
  xlab('F_IS')

Fis_comparison |>
  ggplot(aes(x = x) ) +
  #Top
  geom_histogram(aes(x = Obs, y = ..density..), fill = "#69b3a2") +
  # Bottom
  geom_histogram(aes(x = Exp, y = -..density..), fill = "#404080") +
  theme_bw() +
```

```

xlab('F_IS') +
geom_vline(xintercept = -0.38)

# Turn the per locus basic stats into its own object.
fixation_indices <- locistats_spinosa_Dataset1_Nan[['perloc']]

# Create a new locus column
fixation_indices$Locus <- rownames(fixation_indices)

# Turn into a tibble so it is more predictable
fixation_indices <- tibble(fixation_indices)

# Delete the X from the start of each locus name
fixation_indices$Locus<-gsub("X","",as.character(fixation_indices$Locus))

# Replace the full stop from each locus name with a column.
fixation_indices$Locus<-gsub("[.]",":",as.character(fixation_indices$Locus))

# See if the two files (the one with HWP tests and the one with the Fis values)
# have the same set of loci.
table(fixation_indices$Locus %in% HWreport_spinosa_Dataset1_Nan$Locus)

# Merge...
HWreport_spinosa_Dataset1_Nan.2 <- merge(x = HWreport_spinosa_Dataset1_Nan,
                                         y = fixation_indices, by = 'Locus')

# Plot Fis, marked by significance
HWreport_spinosa_Dataset1_Nan.2 |>
  ggplot(
    aes(x = Fis, y = ..count.., fill = Sig)) +
  geom_histogram() +
  theme_bw()

# Count how many significant outliers are heterozygote excesses and
# how many are deficiencies

# Excesses
count(HWreport_spinosa_Dataset1_Nan.2[Sig == 'sig' & Fis < 0]) / count(
  HWreport_spinosa_Dataset1_Nan.2[Sig == 'sig']) * 100

# Deficiencies
count(HWreport_spinosa_Dataset1_Nan.2[Sig == 'sig' & Fis > 0]) / count(
  HWreport_spinosa_Dataset1_Nan.2[Sig == 'sig']) * 100

# Now look at the whole distribution, is it more negative or more positive?

# Excesses
count(HWreport_spinosa_Dataset1_Nan.2[Fis < -0.1]) / count(
  HWreport_spinosa_Dataset1_Nan.2) * 100

# Fis = 0

```

```
count(HWreport_spinosa_Dataset1_Nan.2[Fis == 0]) / count(
  HWreport_spinosa_Dataset1_Nan.2) * 100

# Deficiencies
count(HWreport_spinosa_Dataset1_Nan.2[Fis > 0.1]) / count(
  HWreport_spinosa_Dataset1_Nan.2) * 100
```

Identity of outlier loci

Now that I have assessed the population as a whole, which loci are outliers? Note that I am currently using a FDR correction based on Benjamini and Hochberg's method of False Discovery Correct. This is to have as much power as possible (because I will be analysing a dataset where I don't exclude any at all), and also to be similar to my global assessment above. It is also the default.

```
# Count the number of significant loci
HWreport_spinosa_Dataset1_Nan %>%
  count(Sig.adj)
```

The loci listed as Sig.adj are recorded as outliers.

spinosa_Dataset1_Oak

Identifying global deviations from HWP

Here I perform a test on all loci to see if the dataset deviates from HWP proportions. I look at Nangur and Oakview separately, because it is a given that they are separate populations and not to be assessed as one.

```
# Perform a test for deviations from HWP
HWreport_spinosa_Dataset1_Oak <- gl.report.hwe(genlight_spinosa_Dataset1_Oak,
  subset = 'all', sig_only = F,
  multi_comp = T, alpha_val = 0.05,
  multi_comp_method = 'fdr')

# Count the number of significant deviations
HWreport_spinosa_Dataset1_Oak %>%
  count(Sig)
```

This is an example of multiple testing. I now need to determine if there are significantly more significant deviations than expected. My critical value is 0.05. I have performed 4659 tests (that is the number of polymorphic loci in this dataset, and they are two-tailed tests. Hence the expected number of type 1 errors (false discovery rate) is: $4659 \times 0.05 = 232.95$.

What follows is a binomial hypothesis test, followed by a visual assessment of the same results, in which p values should be roughly uniformly distributed with no excess of p values below 0.05.

```
# Test if there are significantly more significant deviations than expected
pbinom(q = 222, size = 4437+222, prob = 0.05, lower.tail = FALSE)

# Assess visually.
hist(HWreport_spinosa_Dataset1_Oak$Prob, breaks = 20)
abline(h = 0.05*(4427+222), col = 'red')
```

Assessing nature of global deviations from HWP

One way to investigate further is to assess the F_{IS} distribution and see if it is translated in a positively or negative direction. Negative indicates heterozygote excess (e.g., due to a small N_e), positive indicates heterozygote deficiency (e.g., due to a Wahlund effect). See Waples 2015 for biological processes that can cause these two issues.

```
# Calculate basic stats (particularly  $F_{IS}$ ) for each locus
locistats_spinosa_Dataset1_Oak <- gl.basic.stats(genlight_spinosa_Dataset1_Oak)

# View distribution and summary

# Create dataframe, with both the  $F_{IS}$  values and a null distribution
# (assuming normal with mean of 0 and SD of 0.2).
Fis_comparison <- tibble(
  Obs = na.omit(locistats_spinosa_Dataset1_Oak$perloc$Fis),
  Exp = rnorm(n = length(na.omit(locistats_spinosa_Dataset1_Oak$perloc$Fis)),
    mean = 0, sd = 0.2))

# Plot a histogram showing whether  $F_{IS}$  is translated towards
# positive or negative values.
Fis_comparison |>
  ggplot(aes(x = x) ) +
  #Top
  geom_density(aes(x = Obs, y = ..density..), fill = "#69b3a2") +
  # Bottom
  geom_density(aes(x = Exp, y = -..density..), fill = "#404080") +
  theme_bw() +
  xlab('F_IS')

Fis_comparison |>
  ggplot(aes(x = x) ) +
  #Top
  geom_histogram(aes(x = Obs, y = ..density..), fill = "#69b3a2") +
  # Bottom
  geom_histogram(aes(x = Exp, y = -..density..), fill = "#404080") +
  theme_bw() +
  xlab('F_IS') +
  geom_vline(xintercept = -0.38)

# Turn the per locus basic stats into its own object.
fixation_indices.Oak <- locistats_spinosa_Dataset1_Oak[['perloc']]

# Create a new locus column
fixation_indices.Oak$Locus <- rownames(fixation_indices.Oak)

# Turn into a tibble so it is more predictable
fixation_indices.Oak <- tibble(fixation_indices.Oak)

# Delete the X from the start of each locus name
fixation_indices.Oak$Locus<-gsub("X","",as.character(
  fixation_indices.Oak$Locus))
```

```

# Replace the full stop from each locus name with a column.
fixation_indices.Oak$Locus<-gsub("[.]",":",as.character(
  fixation_indices.Oak$Locus))

# See if the two files (the one with HWP tests and the one with the Fis values)
# have the same set of loci.
table(fixation_indices.Oak$Locus %in% HWreport_spinosa_Dataset1_Oak$Locus)

# Merge...
HWreport_spinosa_Dataset1_Oak.2 <- merge(x = HWreport_spinosa_Dataset1_Oak,
                                         y = fixation_indices.Oak, by = 'Locus')

# Plot Fis, marked by significance
HWreport_spinosa_Dataset1_Oak.2 |>
  ggplot(
    aes(x = Fis, y = ..count..., fill = Sig)) +
  geom_histogram() +
  theme_bw()

# Count how many significant outliers are heterozygote excesses
# and how many are deficiencies

# Excesses
count(HWreport_spinosa_Dataset1_Oak.2[Sig == 'sig' & Fis < 0]) / count(
  HWreport_spinosa_Dataset1_Oak.2[Sig == 'sig']) * 100

# Deficiencies
count(HWreport_spinosa_Dataset1_Oak.2[Sig == 'sig' & Fis > 0]) / count(
  HWreport_spinosa_Dataset1_Oak.2[Sig == 'sig']) * 100

# Now look at the whole distribution, is it more negative or more positive?

# Excesses
count(HWreport_spinosa_Dataset1_Oak.2[Fis < -0.1]) / count(
  HWreport_spinosa_Dataset1_Oak.2) * 100

# Fis = 0
count(HWreport_spinosa_Dataset1_Oak.2[Fis == 0]) / count(
  HWreport_spinosa_Dataset1_Oak.2) * 100

# Deficiencies
count(HWreport_spinosa_Dataset1_Oak.2[Fis > 0.1]) / count(
  HWreport_spinosa_Dataset1_Oak.2) * 100

```

Identity of outlier loci

Now that I have assessed the population as a whole, which loci are outliers? Note that I am currently using a FDR correction based on Benjamini and Hochberg's method of False Discovery Correct. This is to have as much power as possible (because I will be analysing a dataset where I don't exclude any at all), and also to be similar to my global assessment above. It is also the default.

```
# Count the number of significant loci
HWreport_spinosa_Dataset1_Oak %>%
  count(Sig.adj)
```

The loci listed as Sig.adj are recorded as outliers.

spinosa_Dataset1_Oak_wild

Identifying global deviations from HWP

Here I perform a test on all loci to see if the dataset deviates from HWP proportions. I look at Nangur and Oakview separately, because it is a given that they are separate populations and not to be assessed as one.

```
# Perform a test for deviations from HWP
HWreport_spinosa_Dataset1_Oak_wild <- gl.report.hwe(
  genlight_spinosa_Dataset1_Oak_wild,
  subset = 'all', sig_only = F, multi_comp = T, alpha_val = 0.05,
  multi_comp_method = 'fdr')

# Count the number of significant deviations
HWreport_spinosa_Dataset1_Oak_wild %>%
  count(Sig)
```

This is an example of multiple testing. I now need to determine if there are significantly more significant deviations than expected. My critical value is 0.05. I have performed 4656 tests (that is the number of polymorphic loci in this dataset, and they are two-tailed tests. Hence the expected number of type 1 errors (false discovery rate) is: $4656 \times 0.05 = 232.8$.

What follows is a binomial hypothesis test, followed by a visual assessment of the same results, in which p values should be roughly uniformly distributed with no excess of p values below 0.05.

```
# Test if there are significantly more significant deviations than expected
pbinom(q = 229, size = 4427+229, prob = 0.05, lower.tail = FALSE)

# Assess visually.
hist(HWreport_spinosa_Dataset1_Oak_wild$Prob, breaks = 20)
abline(h = 0.05*(4427+229), col = 'red')
```

Assessing nature of global deviations from HWP

One way to investigate further is to assess the Fis distribution and see if it is translated in a positively or negative direction. Negative indicates heterozygote excess (e.g., due to a small Ne), positive indicates heterozygote deficiency (e.g., due to a Wahlund effect). See Waples 2015 for biological processes that can cause these two issues.

```
# Calculate basic stats (particularly Fis) for each locus
locistats_spinosa_Dataset1_Oak_wild <- gl.basic.stats(
  genlight_spinosa_Dataset1_Oak_wild)

# View distribution and summary
```

```

# Create dataframe, with both the Fis values and a null distribution
# (assuming normal with mean of 0 and SD of 0.2).
Fis_comparison <- tibble(
  Obs = na.omit(locistats_spinoso_Dataset1_Oak_wild$perloc$Fis),
  Exp = rnorm(n = length(na.omit(
    locistats_spinoso_Dataset1_Oak_wild$perloc$Fis)), mean = 0, sd = 0.2))

# Plot a histogram showing whether Fis is translated towards
# positive or negative values.
Fis_comparison |>
  ggplot(aes(x = x) ) +
  #Top
  geom_density(aes(x = Obs, y = ..density..), fill = "#69b3a2") +
  # Bottom
  geom_density(aes(x = Exp, y = -..density..), fill = "#404080") +
  theme_bw() +
  xlab('F_IS')

Fis_comparison |>
  ggplot(aes(x = x) ) +
  #Top
  geom_histogram(aes(x = Obs, y = ..density..), fill = "#69b3a2") +
  # Bottom
  geom_histogram(aes(x = Exp, y = -..density..), fill = "#404080") +
  theme_bw() +
  xlab('F_IS') +
  geom_vline(xintercept = -0.38)

```

```

# Turn the per locus basic stats into its own object.
fixation_indices.Oak_wild <- locistats_spinoso_Dataset1_Oak_wild[['perloc']]

# Create a new locus column
fixation_indices.Oak_wild$Locus <- rownames(fixation_indices.Oak_wild)

# Turn into a tibble so it is more predictable
fixation_indices.Oak_wild <- tibble(fixation_indices.Oak_wild)

# Delete the X from the start of each locus name
fixation_indices.Oak_wild$Locus<-gsub("X","",
  as.character(
    fixation_indices.Oak_wild$Locus))

# Replace the full stop from each locus name with a column.
fixation_indices.Oak_wild$Locus<-gsub("[.]",":",
  as.character(
    fixation_indices.Oak_wild$Locus))

# See if the two files (the one with HWP tests and the one with the Fis values)
# have the same set of loci.
table(
  fixation_indices.Oak_wild$Locus %in% HWreport_spinoso_Dataset1_Oak_wild$Locus)

```

```

# Merge...
HWreport_spinosa_Dataset1_Oak_wild.2 <- merge(
  x = HWreport_spinosa_Dataset1_Oak_wild,
  y = fixation_indices.Oak_wild, by = 'Locus')

# Plot Fis, marked by significance
HWreport_spinosa_Dataset1_Oak_wild.2 |>
  ggplot(
    aes(x = Fis, y = ..count.., fill = Sig)) +
  geom_histogram() +
  theme_bw()

# Count how many significant outliers are heterozygote excesses
# and how many are deficiencies

# Excesses
count(HWreport_spinosa_Dataset1_Oak_wild.2[Sig == 'sig' & Fis < 0]) / count(
  HWreport_spinosa_Dataset1_Oak_wild.2[Sig == 'sig']) * 100

# Deficiencies
count(HWreport_spinosa_Dataset1_Oak_wild.2[Sig == 'sig' & Fis > 0]) / count(
  HWreport_spinosa_Dataset1_Oak_wild.2[Sig == 'sig']) * 100

# Now look at the whole distribution, is it more negative or more positive?

# Excesses
count(HWreport_spinosa_Dataset1_Oak_wild.2[Fis < -0.1]) / count(
  HWreport_spinosa_Dataset1_Oak_wild.2) * 100

# Fis = 0
count(HWreport_spinosa_Dataset1_Oak_wild.2[Fis == 0]) / count(
  HWreport_spinosa_Dataset1_Oak_wild.2) * 100

# Deficiencies
count(HWreport_spinosa_Dataset1_Oak_wild.2[Fis > 0.1]) / count(
  HWreport_spinosa_Dataset1_Oak_wild.2) * 100

```

Identity of outlier loci

Now that I have assessed the population as a whole, which loci are outliers? Note that I am currently using a FDR correction based on Benjamini and Hochberg's method of False Discovery Correct. This is to have as much power as possible (because I will be analysing a dataset where I don't exclude any at all), and also to be similar to my global assessment above. It is also the default.

```

# Count the number of significant loci
HWreport_spinosa_Dataset1_Oak_wild %>%
  count(Sig.adj)

```

The loci listed as Sig.adj are recorded as outliers.

spinosa_Dataset1_Nan_FSfilter

Identifying global deviations from HWP

Here I perform a test on all loci to see if the dataset deviates from HWP proportions. I look at Nangur and Oakview separately, because it is a given that they are separate populations and not to be assessed as one.

```
# Perform a test for deviations from HWP
HWreport_spinosa_Dataset1_Nan_FSfilter <- gl.report.hwe(
  genlight_spinosa_Dataset1_Nan_FSfilter, subset = 'all', sig_only = F,
  multi_comp = T, alpha_val = 0.05, multi_comp_method = 'fdr')

# Count the number of significant deviations
HWreport_spinosa_Dataset1_Nan_FSfilter %>%
  count(Sig)
```

This is an example of multiple testing. I now need to determine if there are significantly more significant deviations than expected. My critical value is 0.05. I have performed 2876 tests (that is the number of polymorphic loci in this dataset, and they are two-tailed tests. Hence the expected number of type 1 errors (false discovery rate) is: $2876 \times 0.05 = 143.8$

What follows is a binomial hypothesis test, followed by a visual assessment of the same results, in which p values should be roughly uniformly distributed with no excess of p values below 0.05.

```
# Test if there are significantly more significant deviations than expected
pbinom(q = 88, size = 2788+88, prob = 0.05, lower.tail = FALSE)

# Assess visually.
hist(HWreport_spinosa_Dataset1_Nan_FSfilter$Prob, breaks = 20)
abline(h = 0.05*(2788+88), col = 'red')
```

Assessing nature of global deviations from HWP

One way to investigate further is to assess the Fis distribution and see if it is translated in a positively or negative direction. Negative indicates heterozygote excess (e.g., due to a small N_e), positive indicates heterozygote deficiency (e.g., due to a Wahlund effect). See Waples 2015 for biological processes that can cause these two issues.

```
# Calculate basic stats (particularly Fis) for each locus
locistats_spinosa_Dataset1_Nan_FSfilter <- gl.basic.stats(
  genlight_spinosa_Dataset1_Nan_FSfilter)

# View distribution and summary

hist(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis)

summary(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis)

# Create dataframe, with both the Fis values and a null distribution
# (assuming normal with mean of 0 and SD of 0.2).
Fis_comparison <- tibble(
  value = c(na.omit(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis),
    rnorm(n = length(na.omit(
```

```

        locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis)),
        mean = 0, sd = 0.2)),
type = c(rep(c('Fis', 'null'),
            each = length(
                na.omit(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis))))))

# Plot a histogram showing whether Fis is translated towards positive
# or negative values.
Fis_comparison %>%
  ggplot( aes(x=value, fill=type)) +
  geom_histogram(alpha=0.6, position = 'identity') +
  scale_fill_manual(values=c("black", "grey")) +
  theme_classic() +
  labs(fill="") +
  ggtitle(label = 'plot comparing Fis values to a normal distribution',
          subtitle = paste(
            'mean =',
            mean(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis),
            ', SD =', sd(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis),
            ', median =',
            median(locistats_spinosa_Dataset1_Nan_FSfilter$perloc$Fis)))

```

Identity of outlier loci

Now that I have assessed the population as a whole, which loci are outliers? Note that I am currently using a FDR correction based on Benjamini and Hochberg's method of False Discovery Correct. This is to have as much power as possible (because I will be analysing a dataset where I don't exclude any at all), and also to be similar to my global assessment above. It is also the default.

```

# Count the number of significant loci
HWreport_spinosa_Dataset1_Nan_FSfilter %>%
  count(Sig.adj)

```

The loci listed as Sig.adj are recorded as outliers. There are none.

spinosa_Dataset1_Oak_wild_FSfilter

Identifying global deviations from HWP

Here I perform a test on all loci to see if the dataset deviates from HWP proportions. I look at Nangur and Oakview separately, because it is a given that they are separate populations and not to be assessed as one.

```

# Perform a test for deviations from HWP
HWreport_spinosa_Dataset1_Oak_wild_FSfilter <- gl.report.hwe(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  subset = 'all', sig_only = F, multi_comp = T, alpha_val = 0.05,
  multi_comp_method = 'fdr')

# Count the number of significant deviations
HWreport_spinosa_Dataset1_Oak_wild_FSfilter %>%
  count(Sig)

```

This is an example of multiple testing. I now need to determine if there are significantly more significant deviations than expected. My critical value is 0.05. I have performed 4649 tests (that is the number of polymorphic loci in this dataset, and they are two-tailed tests. Hence the expected number of type 1 errors (false discovery rate) is: $4649 \times 0.05 = 232.45$.

What follows is a binomial hypothesis test, followed by a visual assessment of the same results, in which p values should be roughly uniformly distributed with no excess of p values below 0.05.

```
# Test if there are significantly more significant deviations than expected
pbinom(q = 183, size = 4466+183, prob = 0.05, lower.tail = FALSE)

# Assess visually.
hist(HWreport_spinosa_Dataset1_Oak_wild_FSfilter$Prob, breaks = 20)
abline(h = 0.05*(4466+183), col = 'red')
```

Assessing nature of global deviations from HWP

One way to investigate further is to assess the Fis distribution and see if it is translated in a positively or negative direction. Negative indicates heterozygote excess (e.g., due to a small N_e), positive indicates heterozygote deficiency (e.g., due to a Wahlund effect). See Waples 2015 for biological processes that can cause these two issues.

```
# Calculate basic stats (particularly Fis) for each locus
locistats_spinosa_Dataset1_Oak_wild_FSfilter <- gl.basic.stats(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter)

# View distribution and summary

hist(locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis)

summary(locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis)

# Create dataframe, with both the Fis values and a null distribution
# (assuming normal with mean of 0 and SD of 0.2).
Fis_comparison <- tibble(
  value = c(na.omit(locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis),
    rnorm(n = length(na.omit(
      locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis)),
      mean = 0, sd = 0.2)),
  type = c(rep(c('Fis', 'null'),
    each = length(na.omit(
      locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis))))))

# Plot a histogram showing whether Fis is translated towards positive
# or negative values.
Fis_comparison %>%
  ggplot(aes(x=value, fill=type)) +
  geom_histogram(alpha=0.6, position = 'identity') +
  scale_fill_manual(values=c("black", "grey")) +
  theme_classic() +
  labs(fill="") +
  ggtitle(label = 'plot comparing Fis values to a normal distribution',
    subtitle = paste(
```

```
'mean =' ,
mean(locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis),
', SD =' ,
sd(locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis),
', median =' ,
median(locistats_spinosa_Dataset1_Oak_wild_FSfilter$perloc$Fis)))
```

Identity of outlier loci

Now that I have assessed the population as a whole, which loci are outliers? Note that I am currently using a FDR correction based on Benjamini and Hochberg's method of False Discovery Correct. This is to have as much power as possible (because I will be analysing a dataset where I don't exclude any at all), and also to be similar to my global assessment above. It is also the default.

```
# Count the number of significant loci
HWreport_spinosa_Dataset1_Oak_wild_FSfilter %>%
  count(Sig.adj)
```

The loci listed as Sig.adj are recorded as outliers. There are only 6 outliers (~0.13% of loci).

4. Diversity and Inbreeding

Genetic Diversity and Inbreeding:

The purpose of this analysis is to:

- Determine and compare the level of genetic diversity present in the populations of Oakview and Nangur.
- Generate robust estimates using variant and invariant sites.
- Assess the effect of sample size on my estimates.
- Compare the results of the current population to previous populations using older samples.
- To assess how many individuals would be required to capture 95% of Oakview's diversity.
- Assess genetic diversity at the individual level as a proxy for inbreeding.
- To map the level of inbreeding across both populations.

Population-wide genomic diversity

To assess genetic diversity accurately both variant and invariant sites are required (Schmidt et al., 2021).

The datasets spinosa_Dataset2 and spinosa_Dataset3 are made to contain both variant and invariant markers. See the paper for more information about how they were generated.

To assess the basics of diversity I use the package dartR. See the .adj results for the most robust estimates that factor in both variant and invariant sites.

Comparing MLH through time

Now, as I did with population-wide estimates, I will estimate MLH for a dataset that contains the old samples and the new samples.

```
# Calculate MLH for older samples
MLH_spinosa_Dataset3 <- genlight_spinosa_Dataset3 |>
  gl.report.heterozygosity(
    method = "ind",
    n.invariant = 3507489,
    #plot.display = TRUE, # dartRverse: plot.display = TRUE
  )
```

Merge the old population's results with individual metadata.

```
# Rename ind.name INDV

colnames(MLH_spinosa_Dataset3)[1] <- 'INDV'

# Merge the MLH values with NanguraSpinosa.dat

MLH_spinosa_Dataset3 <- merge(x = MLH_spinosa_Dataset3,
                             y = NanguraSpinosa.dat, by = 'INDV')

# Calculate the IQR for each population
summary(MLH_spinosa_Dataset3$Ho[MLH_spinosa_Dataset3$Population == 'Nangur'])

summary(MLH_spinosa_Dataset3$Ho[MLH_spinosa_Dataset3$Population == 'Oakview'])

# Modify this data frame so that it shows two things...
# 1. a new population that are the captives.
# 2. generation in captivity 1st or 2nd.
MLH_spinosa_Dataset3$Population[
  MLH_spinosa_Dataset3$Sample_clusterK3 == 'captiveborn' |
  MLH_spinosa_Dataset3$Sample_clusterK3 == 'wildborn_captive'] <- 'Captive'

# Reorder the populations so Captive comes next to Oakview.
MLH_spinosa_Dataset3$Population <- factor(MLH_spinosa_Dataset3$Population,
                                           levels = c(
                                             'Nangur', 'Oakview', 'Captive'))

# Under Year, indicate which individuals were born in captivity
MLH_spinosa_Dataset3$Year <- factor(
  MLH_spinosa_Dataset3$Year, levels = c(
    '2006-2007', '2020-2021', 'Born in captivity\n(2020-2021)'))

MLH_spinosa_Dataset3$Year[
  MLH_spinosa_Dataset3$Sample_clusterK3 == 'captiveborn'] <-
  'Born in captivity\n(2020-2021)'
```

```
summary(MLH_spinosa_Dataset3$Ho[MLH_spinosa_Dataset3$Population == 'Captive'])

# For individual inbreeding (F)... based on the max MLH
# See Crow & Kimura (1970)
MLH_spinosa_Dataset3$F_1 <-
  1 - MLH_spinosa_Dataset3$Ho / max(MLH_spinosa_Dataset3$Ho)

# Summarise F
summary(MLH_spinosa_Dataset3$F_1[MLH_spinosa_Dataset3$Population == 'Nangur'])

summary(MLH_spinosa_Dataset3$F_1[MLH_spinosa_Dataset3$Population == 'Oakview'])

summary(MLH_spinosa_Dataset3$F_1[MLH_spinosa_Dataset3$Population == 'Captive'])
```

Because these were output from Populations together, I do not need to adjust for invariant loci etc.
Plot the two datasets together

```
#### Figure 3 ####
MLH_spinosa_Dataset3 |>
  ggplot(

    # Mapping aesthetics to the columns of choice
    aes(fill = Population, x = Population, y = Ho)) +

    # Add violin plots, slightly transparent
    geom_violin(alpha = 0.5) +

    # Plot a boxplot over the top
    geom_boxplot(width=0.1, colour = 'darkgrey') +

    # My favourite theme
    theme_classic() +

    # Y axis label
    ylab('Individual MLH')+

    # Colour fills for violin plots by population
    scale_fill_manual(values = c('forestgreen', 'royalblue', 'royalblue')) +

    # Hide part of legend legend
    guides(fill=FALSE) +

    # Add points and colour by year column
    geom_point(aes(colour = Year)) +

    # Label legend
    labs(colour = 'Year sampled') +
    theme(
      # Embolden axis labels
      axis.title = element_text(face = 2),
      #axis.title.y.right = element_text(angle = 90), # If I want to rotate the axis title

      # Embolden group labels
```

```

axis.text.x = element_text(face = 2)) +
# Colour points
scale_colour_manual(values = c('white', 'black', 'gold')) +
# Put a background on the key so we can see the white circle
theme(legend.key = element_rect(fill = "grey", colour = "white")) +
# Add a second axis, that calculates F based on MLH and the mean MLH of Oakview
scale_y_continuous(name = 'Individual MLH',
                    sec.axis = sec_axis( trans=~./(max(MLH_spinosa_Dataset3$Ho))*(-1)+1, name="F" ) )

```

Checking that heterozygosity is not associated with poorer read depth

There is a chance that the reason the older samples are doing better is not because the population genuinely had more diversity, but because they are poorer quality samples and are biasing the results.

To check this I look at how much read depth explains individual heterozygosity across the dataset using a simple generalised linear model.

Read in the data on individual depth outputted from vcftools.

```

depth_spinosa_Dataset3 <- read.csv(
  file = '../data/other_files/spinosa_Dataset3.idpth', sep = '')

```

Merge depth data with MLH data

```

depth_spinosa_Dataset3 <- merge(x = depth_spinosa_Dataset3,
                                y = MLH_spinosa_Dataset3, by = 'INDV')

```

Run a linear model and plot the results

```

# Run the linear model
model1 <- lm(formula = Ho ~ MEAN_DEPTH ,
              data = depth_spinosa_Dataset3)

# Print the results of the linear model
summary(model1)

# Plot the relationship
depth_spinosa_Dataset3 |>
  ggplot(aes(x = MEAN_DEPTH, y = Ho))+
  geom_point() +
  geom_smooth(method = lm, se = TRUE) +
  theme_classic() +
  xlab('Individual mean depth') +
  ylab('Individual MLH') +
  ylim(0.08,0.18) +
  theme(
    # Embolden axis labels
    axis.title = element_text(face = 2)) +
  annotate('text', x = 40, y = 0.17,
          label =
            paste('R^2 = ',
                  round(summary(model1)$r.squared, 4),

```

```
'\n p = ',
round(summary(model1)[["coefficients"]][2,4], 2)))
```

Table of population-level diversity indices

spinosa_Dataset2

To assess each population I first define the populations in the dataset I am using by the individuals that are found within them.

I also choose to identify the captive-born individuals as a separate population.

```
# Define Oakview population
genlight_spinosa_Dataset2_2 <- gl.define.pop(
  genlight_spinosa_Dataset2,
  ind.list = genlight_spinosa_Dataset2$ind.names[
    genlight_spinosa_Dataset2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], new = 'Oakview')

# Define Nangur population
genlight_spinosa_Dataset2_2 <- gl.define.pop(
  genlight_spinosa_Dataset2_2,
  ind.list = genlight_spinosa_Dataset2_2$ind.names[
    genlight_spinosa_Dataset2_2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], new = 'Nangur')

# Define Captive-born population
genlight_spinosa_Dataset2_2 <- gl.define.pop(
  genlight_spinosa_Dataset2_2,
  ind.list = genlight_spinosa_Dataset2_2$ind.names[
    genlight_spinosa_Dataset2_2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'captive']], new = 'Captive Born')
```

I now assess diversity in these populations.

```
diversity_species_spinosa_Dataset2_2 <- genlight_spinosa_Dataset2_2 %>%
  gl.report.heterozygosity(
    method = "pop",
    n.invariant = 3730497,
    plot.out = TRUE,
  )
```

Fis SD is not reported here. Furthermore, it is found that the FIS results differ from the results gi

```
pa_species_spinosa_Dataset2_2 <- gl.report.pa(genlight_spinosa_Dataset2_2,
  method = 'pairwise')
```

```
#gl.report.diversity(genlight_spinosa_Dataset2_2)
```

I also want to assess the captive population as a whole (including the wild born individuals).

```

# Define Oakview population
genlight_spinosa_Dataset2_3 <- gl.define.pop(
  genlight_spinosa_Dataset2,
  ind.list = genlight_spinosa_Dataset2$ind.names[
    genlight_spinosa_Dataset2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], new = 'Oakview')

# Define Nangur population
genlight_spinosa_Dataset2_3 <- gl.define.pop(
  genlight_spinosa_Dataset2_3,
  ind.list = genlight_spinosa_Dataset2_3$ind.names[
    genlight_spinosa_Dataset2_3$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], new = 'Nangur')

# Define Captive population
genlight_spinosa_Dataset2_3 <- gl.define.pop(
  genlight_spinosa_Dataset2_3,
  ind.list = genlight_spinosa_Dataset2_3$ind.names[
    genlight_spinosa_Dataset2_3$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster2 == 'captiveborn' |
      NanguraSpinosa.dat$Sample_cluster2 == 'wildborn_captive']],
  new = 'Captive_Population')

diversity_species_spinosa_Dataset2_3 <- genlight_spinosa_Dataset2_3 %>%
  gl.report.heterozygosity(
    method = "pop",
    n.invariant = 3730497,
    plot.out = TRUE,
  )

```

I also want to assess the wildborn captive population to the offspring to see how much diversity has been lost through a founder-effect followed by a generation of captive breeding.

```

# Define Oakview population
genlight_spinosa_Dataset2_4 <- gl.define.pop(
  genlight_spinosa_Dataset2,
  ind.list = genlight_spinosa_Dataset2$ind.names[
    genlight_spinosa_Dataset2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], new = 'Oakview')

# Define Nangur population
genlight_spinosa_Dataset2_4 <- gl.define.pop(
  genlight_spinosa_Dataset2_4,
  ind.list = genlight_spinosa_Dataset2_4$ind.names[
    genlight_spinosa_Dataset2_4$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], new = 'Nangur')

# Define Captiveborn population
genlight_spinosa_Dataset2_4 <- gl.define.pop(
  genlight_spinosa_Dataset2_4,
  ind.list = genlight_spinosa_Dataset2_4$ind.names[
    genlight_spinosa_Dataset2_4$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster2 == 'captiveborn']],

```

```

new = 'Captive Born')

# Define wild born, captive population
genlight_spinosa_Dataset2_4 <- gl.define.pop(
  genlight_spinosa_Dataset2_4,
  ind.list = genlight_spinosa_Dataset2_4$ind.names[
    genlight_spinosa_Dataset2_4$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster2 == 'wildborn_captive']],
  new = 'Wild-Born Captive')

diversity_species_spinosa_Dataset2_4 <- genlight_spinosa_Dataset2_4 %>%
  gl.report.heterozygosity(
    method = "pop",
    n.invariant = 3730497,
    plot.out = TRUE,
  )

```

Here I calculate individual multiple-locus heterozygosity values. This is just the proportion of SNPs that are heterozygous. I have not included the old samples as above, instead I am trying to find the current inter-quartile range (IQR) for each population.

```

MLH_spinosa_Dataset2 <- genlight_spinosa_Dataset2 |>
  gl.report.heterozygosity(
    method = "ind",
    n.invariant = 3730497,
    plot.out = TRUE,
  )

```

Merge the results with metadata so that I can view what population they come from and associate other metadata from that individual with its MLH.

```

# Rename ind.name INDV

colnames(MLH_spinosa_Dataset2)[1] <- 'INDV'

# Merge the MLH values with NanguraSpinosa.dat

MLH_spinosa_Dataset2 <- merge(x = MLH_spinosa_Dataset2,
                             y = NanguraSpinosa.dat, by = 'INDV')

# Calculate the IQR for each population
MLHsummNangur <- summary(MLH_spinosa_Dataset2$Ho[
  MLH_spinosa_Dataset2$Population == 'Nangur'])

MLHsummOakview <- summary(MLH_spinosa_Dataset2$Ho[
  MLH_spinosa_Dataset2$Population == 'Oakview' &
  MLH_spinosa_Dataset2$Sample_cluster2 != 'captiveborn'])

# Modify this data frame so that it shows two things...
# 1. a new population that are the captives.
# 2. generation in captivity 1st or 2nd.

```

```

MLH_spinosa_Dataset2$Population[
  MLH_spinosa_Dataset2$Sample_clusterK3 == 'captiveborn' |
  MLH_spinosa_Dataset2$Sample_clusterK3 == 'wildborn_captive'] <- 'Captive'

# Reorder the populations so Captive comes next to Oakview.
MLH_spinosa_Dataset2$Population <- factor(
  MLH_spinosa_Dataset2$Population, levels = c('Nangur', 'Oakview', 'Captive'))

# Under Year, indicate which individuals were born in captivity
MLH_spinosa_Dataset2$Year <- factor(MLH_spinosa_Dataset2$Year, levels = c(
  '2006-2007', '2020-2021', 'Born in captivity'))

MLH_spinosa_Dataset2$Year[
  MLH_spinosa_Dataset2$Sample_clusterK3 == 'captiveborn'] <-
  'Born in captivity'

MLHsummCaptive <- summary(
  MLH_spinosa_Dataset2$Ho[MLH_spinosa_Dataset2$Population == 'Captive'])

```

Estimate F for Dataset 2 (Table)

```

# Estimate F
MLH_spinosa_Dataset2$F_1 <-
  1 - MLH_spinosa_Dataset2$Ho / max(MLH_spinosa_Dataset2$Ho)

# Summarise for Oakview
summary(MLH_spinosa_Dataset2$F_1[
  MLH_spinosa_Dataset2$ind.name %in% MLH_spinosa_Dataset3$INDV[
    MLH_spinosa_Dataset3$Population == 'Oakview' |
    MLH_spinosa_Dataset3$Population == 'Captive' &
    MLH_spinosa_Dataset3$Sample_cluster2 != 'captiveborn'
  ]])

# Summarise for Captives
summary(MLH_spinosa_Dataset2$F_1[
  MLH_spinosa_Dataset2$ind.name %in% MLH_spinosa_Dataset3$INDV[
    MLH_spinosa_Dataset3$Population == 'Captive'
  ]])

# Summarise for Nangur
summary(MLH_spinosa_Dataset2$F_1[
  MLH_spinosa_Dataset2$ind.name %in% MLH_spinosa_Dataset3$INDV[
    MLH_spinosa_Dataset3$Population == 'Nangur'
  ]])

```

Genetic Diversity Table for paper Output the useful results from above into a single dataframe that can be easily transcribed.

```

table2a.headers <- c('Metric', 'Nangur NP', 'Oakview NP', 'Captive Population')

table2a.samplesize <- c(

```

```

'Sample size',
# Nangur
diversity_species_spinosa_Dataset2_2$Ind[2],
# Oakview
diversity_species_spinosa_Dataset2_2$Ind[3],
# Captive Population
diversity_species_spinosa_Dataset2_3$Ind[1]
)

table2a.SNPHobs <- c(
'SNP Hobs',
# Nangur
paste(diversity_species_spinosa_Dataset2_2$Ho[2],
'(',diversity_species_spinosa_Dataset2_2$HoSD[2],')'),
# Oakview
paste(diversity_species_spinosa_Dataset2_2$Ho[3],
'(',diversity_species_spinosa_Dataset2_2$HoSD[3],')'),
# Captive Population
paste(diversity_species_spinosa_Dataset2_3$Ho[1],
'(',diversity_species_spinosa_Dataset2_3$HoSD[1],')')
)

table2a.GWHobs <- c(
'Genome-wide Hobs',
# Nangur
paste(diversity_species_spinosa_Dataset2_2$Ho.adj[2],
'(',diversity_species_spinosa_Dataset2_2$Ho.adjSD[2],')'),
# Oakview
paste(diversity_species_spinosa_Dataset2_2$Ho.adj[3],
'(',diversity_species_spinosa_Dataset2_2$Ho.adjSD[3],')'),
# Captive Population
paste(diversity_species_spinosa_Dataset2_3$Ho.adj[1],
'(',diversity_species_spinosa_Dataset2_3$Ho.adjSD[1],')')
)

table2a.SNPHeXP <- c(
'SNP HeXP',
# Nangur
paste(diversity_species_spinosa_Dataset2_2$He[2],
'(',diversity_species_spinosa_Dataset2_2$HeSD[2],')'),
# Oakview
paste(diversity_species_spinosa_Dataset2_2$He[3],
'(',diversity_species_spinosa_Dataset2_2$HeSD[3],')'),
# Captive Population
paste(diversity_species_spinosa_Dataset2_3$He[1],
'(',diversity_species_spinosa_Dataset2_3$HeSD[1],')')
)

table2a.GWHeXP <- c(
'Genome-wide HeXP',
# Nangur
paste(diversity_species_spinosa_Dataset2_2$He.adj[2],
'(',diversity_species_spinosa_Dataset2_2$He.adjSD[2],')'),

```

```

# Oakview
paste( diversity_species_spinosa_Dataset2_2$He.adj[3],
'(',diversity_species_spinosa_Dataset2_2$He.adjSD[3],')'),
# Captive Population
paste( diversity_species_spinosa_Dataset2_3$He.adj[1],
'(',diversity_species_spinosa_Dataset2_3$He.adjSD[1],')')
)

table2a.FIS <- c(
'FIS',
# Nangur
diversity_species_spinosa_Dataset2_2$FIS[2],
# Oakview
diversity_species_spinosa_Dataset2_2$FIS[3],
# Captive Population
diversity_species_spinosa_Dataset2_3$FIS[1]
)

table2a.MLH_IQR <- c(
'MLH_IQR',
#Nangur
paste(MLHsummNangur[2], '-', MLHsummNangur[5]),
#Oakview
paste(MLHsummOakview[2], '-', MLHsummOakview[5]),
#Captive
paste(MLHsummCaptive[2], '-', MLHsummCaptive[5])
)

table2a.poly <- c(
'%Poly',
# Nangur
diversity_species_spinosa_Dataset2_2$polyLoc[2]/
diversity_species_spinosa_Dataset2_2$nLoc[2]*100,
# Oakview
diversity_species_spinosa_Dataset2_2$polyLoc[3]/
diversity_species_spinosa_Dataset2_2$nLoc[3]*100,
# Captive Population
diversity_species_spinosa_Dataset2_3$polyLoc[1]/
diversity_species_spinosa_Dataset2_3$nLoc[1]*100
)

table2a.Aunique <- c(
'A_unique',
#Nangur
pa_species_spinosa_Dataset2_2$priv1[3]/
diversity_species_spinosa_Dataset2_2$nLoc[2]*100,
#Oakview
pa_species_spinosa_Dataset2_2$priv2[3]/
diversity_species_spinosa_Dataset2_2$nLoc[3]*100,
#Captive
NA
)

```

```

table2a.Afixed <- c(
  'A_fixed',
  #Both
  pa_species_spinosa_Dataset2_2$fixed[3]/
  diversity_species_spinosa_Dataset2_2$nLoc[2]*100,
  # Other cells
  NA,
  NA
)

table2a.metadata <- c(
  'table metadata',
  'Dataset: Nangura spinosa, Full-sibs remain',

  paste('Number of bi-allelic SNPs:',
        diversity_species_spinosa_Dataset2_2$nLoc[2]),

  'Total number of sites (inc. invariant and SNPs): 3,742,811'
)

table2a <- rbind(
  table2a.headers,
  table2a.samplesize,
  table2a.SNPHobs,
  table2a.GWHobs,
  table2a.SNPHexp,
  table2a.GWHexp,
  table2a.FIS,
  table2a.MLH_IQR,
  table2a.poly,
  table2a.Aunique,
  table2a.Afixed,
  table2a.metadata
)

table2a

```

FSfilter

To assess each population I first define the populations in the dataset I am using by the individuals that are found within them.

I also choose to identify the captive-born individuals as a separate population.

```

# Define Oakview population
genlight_spinosa_Dataset2_FSfilter <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter,
  ind.list = genlight_spinosa_Dataset2_FSfilter$ind.names[
    genlight_spinosa_Dataset2_FSfilter$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], new = 'Oakview')

# Define Nangur population

```

```

genlight_spinosa_Dataset2_FSfilter <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter,
  ind.list = genlight_spinosa_Dataset2_FSfilter$ind.names[
    genlight_spinosa_Dataset2_FSfilter$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], new = 'Nangur')

# Define Captive-born population
# no captive born population as they are all related to otehr individuals
# in the dataset.

# Filter out monomorphs (and manually note the new total of invariant loci
# by adding the filtered monomorphs)
genlight_spinosa_Dataset2_FSfilter <- gl.filter.monomorphs(
  genlight_spinosa_Dataset2_FSfilter)

```

I now assess diversity in these populations.

```

diversity_species_spinosa_Dataset2_FSfilter <-
  genlight_spinosa_Dataset2_FSfilter %>%
  gl.report.heterozygosity(
    method = "pop",
    n.invariant = 3730953, # Used to be > 4000000 before refilter 1 and 2
    plot.out = TRUE,
  )

pa_species_spinosa_Dataset2_FSfilter <- gl.report.pa(
  genlight_spinosa_Dataset2_FSfilter, method = 'pairwise')

#gl.report.diversity(genlight_spinosa_Dataset2_FSfilter)

```

I also want to assess the captive population as a whole (including the wild born individuals).

```

# Define Oakview population
genlight_spinosa_Dataset2_FSfilter_2 <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter,
  ind.list = genlight_spinosa_Dataset2_FSfilter$ind.names[
    genlight_spinosa_Dataset2_FSfilter$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], new = 'Oakview')

# Define Nangur population
genlight_spinosa_Dataset2_FSfilter_2 <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter_2,
  ind.list = genlight_spinosa_Dataset2_FSfilter_2$ind.names[
    genlight_spinosa_Dataset2_FSfilter_2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], new = 'Nangur')

# Define Captive population
genlight_spinosa_Dataset2_FSfilter_2 <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter_2,
  ind.list = genlight_spinosa_Dataset2_FSfilter_2$ind.names[
    genlight_spinosa_Dataset2_FSfilter_2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster2 == 'captiveborn' |
      NanguraSpinosa.dat$Sample_cluster2 == 'wildborn_captive']],

```

```

new = 'Captive_Population')

# Filter out monomorphs (and manually note the new total of invariant loci
# by adding the filtered monomorphs)
genlight_spinosa_Dataset2_FSfilter_2 <- gl.filter.monomorphs(
  genlight_spinosa_Dataset2_FSfilter_2)

diversity_species_spinosa_Dataset2_FSfilter_2 <-
  genlight_spinosa_Dataset2_FSfilter_2 %>%
  gl.report.heterozygosity(
    method = "pop",
    n.invariant = 3730953, #Used to be > 4000000 before refilter 1 and 2
    plot.out = TRUE,
  )

```

I also want to assess the wildborn captive population to the offspring to see how much diversity has been lost through a founder-effect followed by a generation of captive breeding.

```

# Define Oakview population
genlight_spinosa_Dataset2_FSfilter_3 <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter,
  ind.list = genlight_spinosa_Dataset2_FSfilter$ind.names[
    genlight_spinosa_Dataset2_FSfilter$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], new = 'Oakview')

# Define Nangur population
genlight_spinosa_Dataset2_FSfilter_3 <- gl.define.pop(
  genlight_spinosa_Dataset2_FSfilter_3,
  ind.list = genlight_spinosa_Dataset2_FSfilter_3$ind.names[
    genlight_spinosa_Dataset2_FSfilter_3$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], new = 'Nangur')

# Define Captiveborn population
#genlight_spinosa_Dataset2_FSfilter_3 <- gl.define.pop(
#genlight_spinosa_Dataset2_FSfilter_3,
#ind.list = genlight_spinosa_Dataset2_FSfilter_3$ind.names[
#  genlight_spinosa_Dataset2_FSfilter_3$ind.names %in% NanguraSpinosa.dat$INDV[
#    NanguraSpinosa.dat$Sample_cluster2 == 'captiveborn']], new = 'captiveborn')

# Define wild born, captive population
#genlight_spinosa_Dataset2_FSfilter_3 <- gl.define.pop(
#genlight_spinosa_Dataset2_FSfilter_3,
#ind.list = genlight_spinosa_Dataset2_FSfilter_3$ind.names[
#  genlight_spinosa_Dataset2_FSfilter_3$ind.names %in% NanguraSpinosa.dat$INDV[
#    NanguraSpinosa.dat$Sample_cluster2 == 'wildborn_captive']],
#  new = 'wildborn_captive')

diversity_species_spinosa_Dataset2_FSfilter_3 <-
  genlight_spinosa_Dataset2_FSfilter_3 %>%
  gl.report.heterozygosity(
    method = "pop",
    n.invariant = 3730953, # Used to be >4000000 before refilter

```

```

    plot.out = TRUE,
  )

```

Here I calculate individual multiple-locus heterozygosity values. This is just the proportion of SNPs that are heterozygous..

```

MLH_spinosa_Dataset2_FSfilter <- genlight_spinosa_Dataset2_FSfilter %>%
  gl.report.heterozygosity(
    method = "ind",
    n.invariant = 3730953, # Used to be >4000000 before refilter
    plot.out = TRUE,
  )

```

Merge the results with metadata so that I can view what population they come from and associate other metadata from that individual with its MLH.

```

# Rename ind.name INDV

colnames(MLH_spinosa_Dataset2_FSfilter)[1] <- 'INDV'

# Merge the MLH values with NanguraSpinosa.dat
MLH_spinosa_Dataset2_FSfilter <- merge(x = MLH_spinosa_Dataset2_FSfilter,
                                       y = NanguraSpinosa.dat, by = 'INDV')

# Calculate the IQR for each population
MLHsummNangur_FSfilter <- summary(MLH_spinosa_Dataset2_FSfilter$Ho[
  MLH_spinosa_Dataset2_FSfilter$Population == 'Nangur'])

MLHsummOakview_FSfilter <- summary(MLH_spinosa_Dataset2_FSfilter$Ho[
  MLH_spinosa_Dataset2_FSfilter$Population == 'Oakview' &
  MLH_spinosa_Dataset2_FSfilter$Sample_cluster2 != 'captiveborn'])

# Modify this data frame so that it shows two things...
# 1. a new population that are the captives.
# 2. generation in captivity 1st or 2nd.
MLH_spinosa_Dataset2_FSfilter$Population[
  MLH_spinosa_Dataset2_FSfilter$Sample_clusterK3 == 'captiveborn' |
  MLH_spinosa_Dataset2_FSfilter$Sample_clusterK3 == 'wildborn_captive'] <-
  'Captive'

# Reorder the populations so Captive comes next to Oakview.
MLH_spinosa_Dataset2_FSfilter$Population <- factor(
  MLH_spinosa_Dataset2_FSfilter$Population,
  levels = c('Nangur', 'Oakview', 'Captive'))

# Under Year, indicate which individuals were born in captivity
MLH_spinosa_Dataset2_FSfilter$Year <- factor(
  MLH_spinosa_Dataset2_FSfilter$Year,
  levels = c('2006-2007', '2020-2021', 'Born in captivity'))

MLH_spinosa_Dataset2_FSfilter$Year[

```

```

MLH_spinosa_Dataset2_FSfilter$Sample_clusterK3 == 'captiveborn'] <-
  'Born in captivity'

MLHsummCaptive_FSfilter <- summary(
  MLH_spinosa_Dataset2_FSfilter$Ho[
    MLH_spinosa_Dataset2_FSfilter$Population == 'Captive'])

```

Genetic Diversity Table for paper Output the useful results from above into a single dataframe that can be easily transcribed.

```

table2c.headers <- c('Metric', 'Nangur NP', 'Oakview NP', 'Captive Population')

table2c.samplesize <- c(
  'Sample size',
  # Nangur
  diversity_species_spinosa_Dataset2_FSfilter$nInd[1],
  # Oakview
  diversity_species_spinosa_Dataset2_FSfilter$nInd[2],
  # Captive Population
  diversity_species_spinosa_Dataset2_FSfilter_2$nInd[1]
)

table2c.SNP Hobs <- c(
  'SNP Hobs',
  # Nangur
  paste(diversity_species_spinosa_Dataset2_FSfilter$Ho[1],
    '(',diversity_species_spinosa_Dataset2_FSfilter$HoSD[1],')'),
  # Oakview
  paste(diversity_species_spinosa_Dataset2_FSfilter$Ho[2],
    '(',diversity_species_spinosa_Dataset2_FSfilter$HoSD[2],')'),
  # Captive Population
  paste(diversity_species_spinosa_Dataset2_FSfilter_2$Ho[1],
    '(',diversity_species_spinosa_Dataset2_FSfilter_2$HoSD[1],')')
)

table2c.GWHobs <- c(
  'Genome-wide Hobs',
  # Nangur
  paste(diversity_species_spinosa_Dataset2_FSfilter$Ho.adj[1],
    '(',diversity_species_spinosa_Dataset2_FSfilter$Ho.adjSD[1],')'),
  # Oakview
  paste(diversity_species_spinosa_Dataset2_FSfilter$Ho.adj[2],
    '(',diversity_species_spinosa_Dataset2_FSfilter$Ho.adjSD[2],')'),
  # Captive Population
  paste(diversity_species_spinosa_Dataset2_FSfilter_2$Ho.adj[1],
    '(',diversity_species_spinosa_Dataset2_FSfilter_2$Ho.adjSD[1],')')
)

table2c.SNP Hexp <- c(
  'SNP Hexp',
  # Nangur
  paste(diversity_species_spinosa_Dataset2_FSfilter$He[1],
    '(',diversity_species_spinosa_Dataset2_FSfilter$HeSD[1],')'),

```

```

# Oakview
paste( diversity_species_spinosa_Dataset2_FSfilter$He[2],
'(',diversity_species_spinosa_Dataset2_FSfilter$HeSD[2],')'),
# Captive Population
paste( diversity_species_spinosa_Dataset2_FSfilter_2$He[1],
'(',diversity_species_spinosa_Dataset2_FSfilter_2$HeSD[1],')')
)

table2c.GWHexp <- c(
'Genome-wide Hexp',
# Nangur
paste( diversity_species_spinosa_Dataset2_FSfilter$He.adj[1],
'(',diversity_species_spinosa_Dataset2_FSfilter$He.adjSD[1],')'),
# Oakview
paste( diversity_species_spinosa_Dataset2_FSfilter$He.adj[2],
'(',diversity_species_spinosa_Dataset2_FSfilter$He.adjSD[2],')'),
# Captive Population
paste( diversity_species_spinosa_Dataset2_FSfilter_2$He.adj[1],
'(',diversity_species_spinosa_Dataset2_FSfilter_2$He.adjSD[1],')')
)

table2c.FIS <- c(
'FIS',
# Nangur
diversity_species_spinosa_Dataset2_FSfilter$FIS[1],
# Oakview
diversity_species_spinosa_Dataset2_FSfilter$FIS[2],
# Captive Population
diversity_species_spinosa_Dataset2_FSfilter_2$FIS[1]
)

table2c.MLH_IQR <- c(
'MLH_IQR',
#Nangur
paste(MLHsummNangur_FSfilter[2], '-', MLHsummNangur_FSfilter[5]),
#Oakview
paste(MLHsummOakview_FSfilter[2], '-', MLHsummOakview_FSfilter[5]),
#Captive
paste(MLHsummCaptive_FSfilter[2], '-', MLHsummCaptive_FSfilter[5])
)

table2c.poly <- c(
'%Poly',
# Nangur
diversity_species_spinosa_Dataset2_FSfilter$polyLoc[1]/
diversity_species_spinosa_Dataset2_FSfilter$nLoc[1]*100,
# Oakview
diversity_species_spinosa_Dataset2_FSfilter$polyLoc[2]/
diversity_species_spinosa_Dataset2_FSfilter$nLoc[2]*100,
# Captive Population
diversity_species_spinosa_Dataset2_FSfilter_2$polyLoc[1]/
diversity_species_spinosa_Dataset2_FSfilter_2$nLoc[1]*100
)

```

```

)

table2c.Aunique <- c(
  'A_unique',
  #Nangur
  pa_species_spinosa_Dataset2_FSfilter$priv1[1]/
  diversity_species_spinosa_Dataset2_FSfilter$nLoc[1]*100,
  #Oakview
  pa_species_spinosa_Dataset2_FSfilter$priv2[1]/
  diversity_species_spinosa_Dataset2_FSfilter$nLoc[2]*100,
  #Captive
  NA
)

table2c.Afixed <- c(
  'A_fixed',
  #Both
  pa_species_spinosa_Dataset2_FSfilter$fixed[1]/
  diversity_species_spinosa_Dataset2_FSfilter$nLoc[1]*100,
  # Other cells
  NA,
  NA
)

table2c.metadata <- c(
  'table metadata',
  'Dataset: Nangura spinosa, Full-sibs purged',

  paste('Number of bi-allelic SNPs:',
        diversity_species_spinosa_Dataset2_FSfilter$nLoc[1]),

  'Total number of sites (inc. invariant and SNPs): 3,730,953'
)

table2c <- rbind(
  table2c.headers,
  table2c.samplesize,
  table2c.SNPHobs,
  table2c.GWHobs,
  table2c.SNPHexp,
  table2c.GWHexp,
  table2c.FIS,
  table2c.MLH_IQR,
  table2c.poly,
  table2c.Aunique,
  table2c.Afixed,
  table2c.metadata
)

table2c

```

Genetic Capture

One of the managers' goals is to establish an insurance population at another site. For this to be successful from a genetic perspective they need to initially capture >95% of the diversity of the source population (Weeks et al., 2011), and then the N_e needs to be rapidly increased to 1000 in order to retain >90% of the necessary diversity. Individuals sampled will need to be of equal numbers of each sex, from a diverse range of localities, and need to be able to contribute equally to each generation. Continued supplementation from the source population will be necessary.

I will use the random sampling 'bootstrap' approach from above to simulate sampling Oakview at several different sample sizes. This will demonstrate how many individuals are required to confidently (>95% of cases) capture >95% of Oakview's diversity.

To save time and recreate the main plot using the data I used, skip to the sub-heading 'Short-cut: use saved data'.

Oakview

```
# Number of replicates / sample size
nrep <- 1:100

# Sample sizes to test (4 - 20, by steps of 4)
sample.size <- c(seq(from = 5, to = 45, by = 5))

# Create a table that has the above info
results_capture <- expand.grid(nrep = nrep, sample.size = sample.size)

# I am only interested in Oakview for this analysis,
# and it is not possible to do for Nangur,
temp.gl <- gl.keep.pop(genlight_spinosa_Dataset2_2, pop.list = 'Oakview')

# A for loop that loops through the above table
for (i in 1:nrow(results_capture)) {
  # Print where the for loop is up to
  print(paste0('Loop number: ', i, ' of: ', length(results_capture$nrep)))

  # Subsample a number of individuals absed on the table
  dummy <- gl.sample( temp.gl, nsample = results_capture$sample.size[i],
                      replace = FALSE, verbose = 0)
  # Get that genlight working for the subsample
  dummy <- gl.compliance.check(dummy, verbose = 0)
  #Report diversity metrics for the subsample
  report <- gl.report.heterozygosity(dummy, plot.out = FALSE,
                                     n.invariant = 3730497, verbose = 0)

  # Save results for the correct population
  results_capture$polyLoc[i] <- report$polyLoc/report$nLoc*100
  results_capture$Ho[i] <- report$Ho
  results_capture$He[i] <- report$He
  results_capture$uHe[i] <- report$uHe
  results_capture$Fis[i] <- report$FIS
}
```

```
# Adjust the polymorphic loci results to account for the Oakview population
# rather than the dataset as a whole
```

```
results_capture$polyLoc_Pop <- results_capture$polyLoc/71.72324*100
```

Nangur

```
# Number of replicates / sample size
```

```
nrep <- 1:100
```

```
# Sample sizes to test (4 - 20, by steps of 4)
```

```
sample.size <- c(seq(from = 5, to = 25, by = 5))
```

```
# Create a table that has the above info
```

```
results_capture2 <- expand.grid(nrep = nrep, sample.size = sample.size)
```

```
# Isolate just Nangur
```

```
temp.gl <- gl.keep.pop(genlight_spinosa_Dataset2_2, pop.list = 'Nangur')
```

```
# A for loop that loops through the above table
```

```
for (i in 1:nrow(results_capture2)) {
```

```
  # Print where the for loop is up to
```

```
  print(paste0('Loop number: ', i, ' of: ', length(results_capture2$nrep)))
```

```
  # Subsample a number of individuals absed on the table
```

```
  dummy <- gl.sample( temp.gl, nsample = results_capture2$sample.size[i],
    replace = FALSE, verbose = 0)
```

```
  # Get that genlight working for the subsample
```

```
  dummy <- gl.compliance.check(dummy, verbose = 0)
```

```
  #Report diversity metrics for the subsample
```

```
  report <- gl.report.heterozygosity(dummy, plot.out = FALSE,
    n.invariant = 3730497, verbose = 0)
```

```
  # Save results for the correct population
```

```
  results_capture2$polyLoc[i] <- report$polyLoc/report$nLoc*100
```

```
  results_capture2$Ho[i] <- report$Ho
```

```
  results_capture2$He[i] <- report$He
```

```
  results_capture2$uHe[i] <- report$uHe
```

```
  results_capture2$Fis[i] <- report$FIS
```

```
}
```

```
# Adjust the polymorphic loci results to account for the
```

```
# Nangur population rather than the dataset as a whole
```

```
results_capture2$polyLoc_Pop <- results_capture2$polyLoc/41.33507*100
```

Both (private and fixed alleles)

```
# Number of replicates / sample size
nrep <- 1:100

# Sample sizes to test (4 - 20, by steps of 4)
sample.size <- c(seq(from = 5, to = 25, by = 5))

# Create a table that has the above info
results_capture3 <- expand.grid(nrep = nrep, sample.size = sample.size)

# Isolate just Nangur
temp.gl <- gl.keep.pop(genlight_spinosa_Dataset2_2,
                      pop.list = c('Nangur', 'Oakview'))

# A for loop that loops through the above table
for (i in 1:nrow(results_capture2)) {
  # Print where the for loop is up to
  print(paste0('Loop number: ', i, ' of: ', length(results_capture3$nrep)))

  # Subsample a number of individuals based on the table
  dummy <- gl.sample( temp.gl, nsample = results_capture3$sample.size[i],
                     replace = FALSE, verbose = 0)
  # Get that genlight working for the subsample
  dummy <- gl.compliance.check(dummy, verbose = 0)
  # Report diversity metrics for the subsample
  report <- gl.report.pa(dummy, method = 'pairwise',
                       plot.out = FALSE, verbose = 0)

  # Save results for the correct population
  results_capture3$PrivNan[i] <- report$priv1
  results_capture3$PrivOak[i] <- report$priv2
  results_capture3$fixed[i] <- report$fixed
}
```

Combine the dataframes

I want to plot the results of these analyses together in the same figure.

```
# Add a column that names the population each dataset came from.
results_capture$POP <- 'Oakview'

results_capture2$POP <- 'Nangur'

# Bind the two dataframes together
results_capture.joined <- rbind(results_capture, results_capture2)
```

Short-cut: use saved data

To save time and recreate the main plot using the data I used, run the following chunk before plotting.

```
#results_capture.joined <- read.csv(  
# file = '../data/other_files/genetic_capture_joined_tab.csv')
```

Plot the results

```
# Use spreadsheet genetic_capture_joined_tab.csv for consistent results  
results_capture.joined[results_capture.joined$sample.size != 47,] %>%  
  ggplot(aes(y = as.factor(sample.size), x=polyLoc_Pop, fill = POP)) +  
    geom_density_ridges(alpha = 0.6,  
                        colour = 'black',  
                        quantile_lines = TRUE,  
                        quantiles = c(0.05, 0.5, 0.95))+  
  theme_classic() +  
  scale_fill_manual(values = c('forestgreen', 'royalblue')) +  
  scale_x_continuous('Polymorphic loci of population captured (%)',  
                     breaks = c(50, 60, 70, 80, 90, 95, 100))+  
  scale_y_discrete('Size of subsample') +  
  geom_point(aes(colour = POP), shape = '|', size = 2) +  
  scale_colour_manual(values = c('darkgreen', 'darkblue')) +  
  theme(  
    # Make axis titles bold  
    axis.title = element_text(face = 2)) +  
  # Rename legend  
  labs(fill = 'Population') +  
  guides(colour = FALSE)
```

Supplementary: Effect of sample size

The effect of sample size can be prominent in genetic studies. Here I demonstrate Oakview and Nangur's results are robust to changes in sample size using the datasets generated above. As these subsamples come from the same genlights, an assumption is that a consistent set of loci is being analysed across all subsamples, rather than a new set of loci generated for each subsample.

```
p1 <- results_capture.joined[results_capture.joined$sample.size != 47,] %>%  
  ggplot(aes(y = as.factor(sample.size), x=Ho, fill = POP)) +  
    geom_density_ridges(alpha = 0.6,  
                        colour = 'black',  
                        quantile_lines = TRUE,  
                        quantiles = c(0.05, 0.5, 0.95))+  
  theme_classic() +  
  scale_fill_manual(values = c('forestgreen', 'royalblue')) +  
  scale_x_continuous('Observed heterozygosity')+  
  scale_y_discrete('Size of subsample') +  
  geom_point(aes(colour = POP), shape = '|', size = 2) +
```

```

scale_colour_manual(values = c('lightgreen', 'darkblue')) +
theme(
  # Make axis titles bold
  axis.title = element_text(face = 2)) +
# Rename legend
labs(fill = 'Population') +
guides(colour = FALSE)

p2 <- results_capture.joined[results_capture.joined$sample.size != 47,] %>%
ggplot(aes(y = as.factor(sample.size), x=He, fill = POP)) +
  geom_density_ridges(alpha = 0.6,
    colour = 'black',
    quantile_lines = TRUE,
    quantiles = c(0.05, 0.5, 0.95))+
  theme_classic() +
  scale_fill_manual(values = c('forestgreen', 'royalblue')) +
  scale_x_continuous('Expected heterozygosity')+
  scale_y_discrete('Size of subsample') +
  geom_point(aes(colour = POP), shape = '|', size = 2) +
  scale_colour_manual(values = c('lightgreen', 'darkblue')) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
# Rename legend
labs(fill = 'Population') +
guides(colour = FALSE)

p3 <- results_capture.joined[results_capture.joined$sample.size != 47,] %>%
ggplot(aes(y = as.factor(sample.size), x=uHe, fill = POP)) +
  geom_density_ridges(alpha = 0.6,
    colour = 'black',
    quantile_lines = TRUE,
    quantiles = c(0.05, 0.5, 0.95))+
  theme_classic() +
  scale_fill_manual(values = c('forestgreen', 'royalblue')) +
  scale_x_continuous('Unbiased expected heterozygosity')+
  scale_y_discrete('Size of subsample') +
  geom_point(aes(colour = POP), shape = '|', size = 2) +
  scale_colour_manual(values = c('lightgreen', 'darkblue')) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
# Rename legend
labs(fill = 'Population') +
guides(colour = FALSE)

p4 <- results_capture.joined[results_capture.joined$sample.size != 47,] %>%
ggplot(aes(y = as.factor(sample.size), x=Fis, fill = POP)) +
  geom_density_ridges(alpha = 0.6,

```

```

        colour = 'black',
        quantile_lines = TRUE,
        quantiles = c(0.05, 0.5, 0.95))+
theme_classic() +
scale_fill_manual(values = c('forestgreen', 'royalblue')) +
scale_x_continuous(bquote( F[IS]))+
scale_y_discrete('Size of subsample') +
geom_point(aes(colour = POP), shape = '|', size = 2) +
scale_colour_manual(values = c('lightgreen', 'darkblue')) +
theme(
  # Make axis titles bold
  axis.title = element_text(face = 2)) +
# Rename legend
labs(fill = 'Population') +
guides(colour = FALSE)

p5 <- results_capture.joined[results_capture.joined$sample.size != 47,] %>%
ggplot(aes(y = as.factor(sample.size), x=polyLoc, fill = POP)) +
  geom_density_ridges(alpha = 0.6,
    colour = 'black',
    quantile_lines = TRUE,
    quantiles = c(0.05, 0.5, 0.95))+
theme_classic() +
scale_fill_manual(values = c('forestgreen', 'royalblue')) +
scale_x_continuous('Polymorphic loci (%)')+
scale_y_discrete('Size of subsample') +
geom_point(aes(colour = POP), shape = '|', size = 2) +
scale_colour_manual(values = c('lightgreen', 'darkblue')) +
theme(
  # Make axis titles bold
  axis.title = element_text(face = 2)) +
# Rename legend
labs(fill = 'Population') +
guides(colour = FALSE)

```

Diversity

```

# Number of replicates / sample size
nrep <- 1:100

# Sample sizes to test (4 - 20, by steps of 4)
sample.size <- c(seq(from = 5, to = 25, by = 5))

# Create a table that has the above info
results_capture3 <- expand.grid(nrep = nrep, sample.size = sample.size)

# Isolate just Nangur
temp.gl <- gl.keep.pop(genlight_spinosa_Dataset2_2,

```

```

pop.list = c('Nangur','Oakview'))

# A for loop that loops through the above table
for (i in 1:nrow(results_capture3)) {
  # Print where the for loop is up to
  print(paste0('Loop number: ', i, ' of: ', length(results_capture3$nrep)))

  # Subsample a number of individuals abased on the table
  dummy <- gl.sample( temp.gl, nsample = results_capture3$sample.size[i],
                      replace = FALSE, verbose = 0)
  # Get that genlight working for the subsample
  dummy <- gl.compliance.check(dummy, verbose = 0)
  #Report diversity metrics for the subsample
  report <- gl.report.pa(dummy, method = 'pairwise',
                        plot.out = FALSE, verbose = 0)

  # Save results for the correct population
  results_capture3$Nangur[i] <- report$priv1
  results_capture3$Oakview[i] <- report$priv2
  results_capture3$fixed[i] <- report$fixed

}

# I want the populaton information to be kept in one column
# so they can be plotted easily
results_capture3.long <- results_capture3 |>
  pivot_longer(cols = c(Nangur, Oakview),
               names_to = 'POP',
               values_to = 'private')

# Rename the populations to just Nangur and Oak

```

private and fixed alleles Plot results...

```

# Fixed alleles
p6 <- results_capture3 %>%
  ggplot(aes(y = as.factor(sample.size), x=fixed/12314*100)) +
  geom_density_ridges(alpha = 0.6,
                     colour = 'black',
                     fill = 'grey',
                     quantile_lines = TRUE,
                     quantiles = c(0.05, 0.5, 0.95))+

  theme_classic() +
  scale_x_continuous('SNPs with fixed alleles (%)')+
  scale_y_discrete('Size of subsample') +
  geom_point( shape = '|', size = 2) +
  scale_colour_manual(values = c('lightgreen', 'darkblue')) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
  # Rename legend
  labs(fill = 'Population') +

```

```

guides(colour = FALSE)

# Private alleles
p7 <- results_capture3.long |>
ggplot(aes(y = as.factor(sample.size), x=private/12314*100, fill = POP)) +
  geom_density_ridges(alpha = 0.6,
    colour = 'black',
    quantile_lines = TRUE,
    quantiles = c(0.05, 0.5, 0.95),
    scale = 0.9)+

  theme_classic() +
  scale_fill_manual(values = c('forestgreen', 'royalblue')) +
  scale_x_continuous('SNPs with alleles unique to populations (%)')+
  scale_y_discrete('Size of subsample') +
  geom_point(aes(colour = POP), shape = '|', size = 2) +
  scale_colour_manual(values = c('lightgreen', 'darkblue')) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
  # Rename legend
  labs(fill = 'Population') +
  guides(colour = FALSE)

```

5. AMOVA and Fst

Connectivity and Structure: Other: AMOVA Private Alleles AFD Fst

The purpose of these analyses are to:

- Further characterise the level of connectivity and structure within and between Oakview and Nangur.
- Perform a simple AMOVA to determine what percentage variation various groupings explain.
- Count and visualise the number of private alleles in different groupings, to determine the relative level of connectivity.
- Calculate and compare AFD and Fst, and further assess the relative level of connectivity.

AMOVA

AMOVA (Analysis of Molecular Variance) is a method of analysis that determines how well a priori groups explain variance in a dataset. A full AMOVA is able to determine how much variation exists within and between groups and within and between individuals.

The reason I am performing this is to demonstrate the relative importance of various groupings of individuals that have been produced by other connectivity and structure analyses.

In panmictic populations, most variation arises from within individuals. If there is variation between individuals, then there is unexplained population structure. If there is variation explained by the variables I provide it then that will be important.

Species Level

Here I assess how much variation in the data is accounted for by the major population groupings: Oakview and Nangur. This will be a good comparison for assessing smaller levels.

spinosa_Dataset1 First I need to assign populations to the correct dataset (one that contains both populations and is filtered fully)

```
# Define Oakview
genlight_spinosa_Dataset1_wild_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_wild,
  ind.list = genlight_spinosa_Dataset1_wild$ind.names[
    genlight_spinosa_Dataset1_wild$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Oakview']], 'Oakview')

# Define Nangur
genlight_spinosa_Dataset1_wild_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_wild_K2,
  ind.list = genlight_spinosa_Dataset1_wild_K2$ind.names[
    genlight_spinosa_Dataset1_wild_K2$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Population == 'Nangur']], 'Nangur')

# Create a strata in the genlight representing populations
strata(genlight_spinosa_Dataset1_wild_K2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_wild_K2$pop)
```

Performing the AMOVA using the ade4 method in poppr

```
# Perform amova
amova_spinosa_Dataset1 <- genlight_spinosa_Dataset1_wild_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  #Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Perform ade4 permutation testing
amova_spinosa_Dataset1_sig <- randtest(xtest = amova_spinosa_Dataset1,
  nrepet = 1000)

# Results of AMOVA
amova_spinosa_Dataset1
# Phi stats from AMOVA
amova_spinosa_Dataset1$statphi

# Significance testing of AMOVA
amova_spinosa_Dataset1_sig

# Visualised significance testing of AMOVA
plot(amova_spinosa_Dataset1_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_wild_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
```

```

method = 'pegas' # Slower but more carefully developed than ade4
)

```

spinosa_Dataset1_FSfilter First I need to assign populations to the correct dataset (one that contains both populations and is filtered fully (including siblings))

```

# Define Oakview
genlight_spinosa_Dataset1_wild_FSfilter_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_wild_FSfilter,
  ind.list = genlight_spinosa_Dataset1_wild_FSfilter$ind.names[
    genlight_spinosa_Dataset1_wild_FSfilter$ind.names %in%
      NanguraSpinosa.dat$INDV[NanguraSpinosa.dat$Population == 'Oakview']],
  'Oakview')

# Define Nangur
genlight_spinosa_Dataset1_wild_FSfilter_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_wild_FSfilter_K2,
  ind.list = genlight_spinosa_Dataset1_wild_FSfilter_K2$ind.names[
    genlight_spinosa_Dataset1_wild_FSfilter_K2$ind.names %in%
      NanguraSpinosa.dat$INDV[NanguraSpinosa.dat$Population == 'Nangur']],
  'Nangur')

# Create a strata in the genlight representing populations
strata(genlight_spinosa_Dataset1_wild_FSfilter_K2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_wild_FSfilter_K2$pop)

```

Performing the AMOVA using poppr

```

# Perform amova
amova_spinosa_Dataset1_wild_FSfilter <-
  genlight_spinosa_Dataset1_wild_FSfilter_K2 |>
  poppr.amova(hier = ~Population, # Formula for what strata to analyse
    within = TRUE, # Whether to look within individuals for variation.
    #Must be set to FALSE for pegas implementation of
    # permutation testing (for missing data)
    method = 'ade4'
  )

# Perform ade4 permutation testing
amova_spinosa_Dataset1_wild_FSfilter_sig <- randtest(xtest =
  amova_spinosa_Dataset1_wild_FSfilter, nrepet = 1000)

# Results of AMOVA
amova_spinosa_Dataset1_wild_FSfilter

# Significance testing results
amova_spinosa_Dataset1_wild_FSfilter_sig
# Visualsied significance testing
plot(amova_spinosa_Dataset1_wild_FSfilter_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_wild_FSfilter_K2 |>

```

```
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)
```

Oakview K=2

spinosa_Dataset1_Oak Here I use the DAPC K=2 groupings to see how important that is. Groupings are East and Central. I will exclude the Western individual for this analysis.

```
# Remove the far west individual.
genlight_spinosa_Dataset1_Oak_wild_K2 <- gl.drop.ind(
  genlight_spinosa_Dataset1_Oak_wild,
  ind.list = genlight_spinosa_Dataset1_Oak_wild$ind.names[
    genlight_spinosa_Dataset1_Oak_wild$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'Oakview_farwest']]

# Define a individuals as being in the East population
# based on their sampling cluster in the metadata dataframe
genlight_spinosa_Dataset1_Oak_wild_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_Oak_wild_K2,
  ind.list = genlight_spinosa_Dataset1_Oak_wild_K2$ind.names[
    genlight_spinosa_Dataset1_Oak_wild_K2$ind.names %in%
      NanguraSpinosa.dat$INDV[NanguraSpinosa.dat$Sample_cluster ==
        'Oakview_fareast']], new = 'East')

# Rename the other population 'Central'
genlight_spinosa_Dataset1_Oak_wild_K2 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Oak_wild_K2, old = 'pop1', new = 'Central')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Oak_wild_K2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Oak_wild_K2$pop)
```

Performing the AMOVA using poppr

```
# Perform AMOVA
amova_spinosa_Dataset1_Oak_wild_K2 <- genlight_spinosa_Dataset1_Oak_wild_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Oak_wild_K2_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Oak_wild_K2, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Oak_wild_K2
```

```

# Significance results
amova_spinosa_Dataset1_Oak_wild_K2_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Oak_wild_K2_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Oak_wild_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)

```

spinosa_Dataset1_Oak_FSfilter Here I use the DAPC K=2 groupings to see how important that is. Groupings are East and Central. I will exclude the Western individual for this analysis.

```

# Remove the far west individual.
genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2 <- gl.drop.ind(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  ind.list = genlight_spinosa_Dataset1_Oak_wild_FSfilter$ind.names[
    genlight_spinosa_Dataset1_Oak_wild_FSfilter$ind.names %in%
    NanguraSpinosa.dat$INDV[NanguraSpinosa.dat$Sample_cluster ==
      'Oakview_farwest']]
)

# Define a individuals as being in the East population based on
# their sampling cluster in the metadata dataframe
genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2,
  ind.list = genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2$ind.names[
    genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2$ind.names %in%
    NanguraSpinosa.dat$INDV[NanguraSpinosa.dat$Sample_cluster ==
      'Oakview_fareast']], new = 'East')

# Rename the other population 'Central'
genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2, old = 'pop1',
  new = 'Central')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2$pop)

```

Performing the AMOVA using poppr

```

# Perform AMOVA
amova_spinosa_Dataset1_Oak_wild_FSfilter_K2 <-
  genlight_spinosa_Dataset1_Oak_wild_FSfilter_K2 |>
  poppr.amova(hier = ~Population, # Formula for what strata to analyse

```

```

    within = TRUE, # Whether to look within individuals for variation.
    # Must be set to FALSE for pegas implementation of
    # permutation testing (for missing data)
    method = 'ade4'
  )
# Significance test amova using permutation testing ade4 package
amova_spinoso_Dataset1_Oak_wild_FSfilter_K2_sig <- randtest(
  xtest = amova_spinoso_Dataset1_Oak_wild_FSfilter_K2, nrepet = 1000)

# AMOVA results
amova_spinoso_Dataset1_Oak_wild_FSfilter_K2

# Significance results
amova_spinoso_Dataset1_Oak_wild_FSfilter_K2_sig

# Visualise significance
plot(amova_spinoso_Dataset1_Oak_wild_FSfilter_K2_sig)

# Compare with pegas to double check validity of results
genlight_spinoso_Dataset1_Oak_wild_FSfilter_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)

```

Oakview K=3

`spinoso_Dataset1_Oak_wild` Here I use the DAPC K=3 groupings to see how important that is. Groupings are East and Central and North. I will exclude the Western individual for this analysis.

```

# Remove the far west individual.
genlight_spinoso_Dataset1_Oak_wild_K3 <- gl.drop.ind(
  genlight_spinoso_Dataset1_Oak_wild,
  ind.list = genlight_spinoso_Dataset1_Oak_wild$ind.names[
    genlight_spinoso_Dataset1_Oak_wild$ind.names %in%
    NanguraSpinoso.dat$INDV[
      NanguraSpinoso.dat$Sample_cluster == 'Oakview_fareast']]
)

# Define a individuals as being in the East population based on
# their sampling cluster in the metadata dataframe
genlight_spinoso_Dataset1_Oak_wild_K3 <- gl.define.pop(
  genlight_spinoso_Dataset1_Oak_wild_K3,
  ind.list = genlight_spinoso_Dataset1_Oak_wild_K3$ind.names[
    genlight_spinoso_Dataset1_Oak_wild_K3$ind.names %in%
    NanguraSpinoso.dat$INDV[
      NanguraSpinoso.dat$Sample_cluster == 'Oakview_fareast']], new = 'East')

# Rename the other population 'Central'
genlight_spinoso_Dataset1_Oak_wild_K3 <- gl.rename.pop(

```

```

genlight_spinosa_Dataset1_Oak_wild_K3, old = 'pop1', new = 'Central')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Oak_wild_K3) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Oak_wild_K3$pop)

```

Performing the AMOVA using poppr

```

# Perform AMOVA
amova_spinosa_Dataset1_Oak_wild_K3 <- genlight_spinosa_Dataset1_Oak_wild_K3 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  #Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Oak_wild_K3_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Oak_wild_K3, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Oak_wild_K3

# Significance results
amova_spinosa_Dataset1_Oak_wild_K3_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Oak_wild_K3_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Oak_wild_K3 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)

```

spinosa_Dataset1_Oak_wild_FSfilter Here I use the DAPC K=3 groupings to see how important that is.

Groupings are East and Central and North. I will exclude the Western individual for this analysis.

```

# Remove the far west individual.
genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3 <- gl.drop.ind(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  ind.list = genlight_spinosa_Dataset1_Oak_wild_FSfilter$ind.names[
    genlight_spinosa_Dataset1_Oak_wild_FSfilter$ind.names %in%
      NanguraSpinoso.dat$INDV[NanguraSpinoso.dat$Sample_cluster ==
        'Oakview_farwest']]

# Define a individuals as being in the East population based on their sampling cluster in the metadata

```

```

genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3 <- gl.define.pop(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3,
  ind.list = genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3$ind.names[
    genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3$ind.names %in%
      NanguraSpinosa.dat$INDV[NanguraSpinosa.dat$Sample_cluster ==
        'Oakview_fareast']], new = 'East')

# Rename the other population 'Central'
genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3, old = 'pop1', new = 'Central')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3$pop)

```

Performing the AMOVA using poppr

```

# Perform AMOVA
amova_spinosa_Dataset1_Oak_wild_FSfilter_K3 <-
  genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Oak_wild_FSfilter_K3_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Oak_wild_FSfilter_K3, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Oak_wild_FSfilter_K3

# Significance results
amova_spinosa_Dataset1_Oak_wild_FSfilter_K3_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Oak_wild_FSfilter_K3_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Oak_wild_FSfilter_K3 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)

```

Nangur K=2 (v1. East/West)

spinosa_Dataset1_Nan Here I use the DAPC K=2 groupings to see how important that is.

Groupings are East and West.

```
# Define a individuals as being in the East population based on their
# sampling cluster in the metadata dataframe
genlight_spinosa_Dataset1_Nan_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_Nan,
  ind.list = genlight_spinosa_Dataset1_Nan$ind.names[
    genlight_spinosa_Dataset1_Nan$ind.names %in% NanguraSpinosa.dat$INDV[
      (NanguraSpinosa.dat$Sample_cluster == 'Nangur_fareast' |
        NanguraSpinosa.dat$Sample_cluster == 'Nangur_mideast') &
        NanguraSpinosa.dat$INDV != 'BSU0532']], new = 'East')

# Rename the other population 'Central'
genlight_spinosa_Dataset1_Nan_K2 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Nan_K2, old = 'pop1', new = 'West')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Nan_K2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Nan_K2$pop)
```

Performing the AMOVA using poppr

```
# Perform AMOVA
amova_spinosa_Dataset1_Nan_K2 <- genlight_spinosa_Dataset1_Nan_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  #Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Nan_K2_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Nan_K2, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Nan_K2

# Significance results
amova_spinosa_Dataset1_Nan_K2_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Nan_K2_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Nan_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  #Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)
```

spinosa_Dataset1_Nan_FSfilter Here I use the DAPC K=2 groupings to see how important that is.

Groupings are East and West.

```
# Define a individuals as being in the East population based on their
# sampling cluster in the metadata dataframe
genlight_spinosa_Dataset1_Nan_FSfilter_K2 <- gl.define.pop(
  genlight_spinosa_Dataset1_Nan_FSfilter,
  ind.list = genlight_spinosa_Dataset1_Nan_FSfilter$ind.names[
    genlight_spinosa_Dataset1_Nan_FSfilter$ind.names %in%
      NanguraSpinosa.dat$INDV[(NanguraSpinosa.dat$Sample_cluster ==
        'Nangur_fareast' |
        NanguraSpinosa.dat$Sample_cluster ==
        'Nangur_mideast') &
        NanguraSpinosa.dat$INDV != 'BSU0532']],
  new = 'East')

# Rename the other population 'Central'
genlight_spinosa_Dataset1_Nan_FSfilter_K2 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Nan_FSfilter_K2, old = 'pop1', new = 'West')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Nan_FSfilter_K2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Nan_FSfilter_K2$pop)
```

Performing the AMOVA using poppr

```
# Perform AMOVA
amova_spinosa_Dataset1_Nan_FSfilter_K2 <- genlight_spinosa_Dataset1_Nan_FSfilter_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Nan_FSfilter_K2_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Nan_FSfilter_K2, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Nan_FSfilter_K2

# Significance results
amova_spinosa_Dataset1_Nan_FSfilter_K2_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Nan_FSfilter_K2_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Nan_FSfilter_K2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
```

```
method = 'pegas' # Slower but more carefully developed than ade4
)
```

Nangur K=2 (v2. Far East/West)

spinosa_Dataset1_Nan Here I use the DAPC K=2 groupings to see how important that is.

Groupings are Far East and West.

```
# Define a individuals as being in the East population based on their sampling cluster in the metadata
genlight_spinosa_Dataset1_Nan_K2_v2 <- gl.define.pop(
  genlight_spinosa_Dataset1_Nan,
  ind.list = genlight_spinosa_Dataset1_Nan$ind.names[
    genlight_spinosa_Dataset1_Nan$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'Nangur_fareast']],
  new = 'Far East')

# Rename the other population 'Central'
genlight_spinosa_Dataset1_Nan_K2_v2 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Nan_K2_v2, old = 'pop1', new = 'West')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Nan_K2_v2) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Nan_K2_v2$pop)
```

Performing the AMOVA using poppr

```
# Perform AMOVA
amova_spinosa_Dataset1_Nan_K2_v2 <- genlight_spinosa_Dataset1_Nan_K2_v2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  # Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'ade4'
)

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Nan_K2_v2_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Nan_K2_v2, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Nan_K2_v2

# Significance results
amova_spinosa_Dataset1_Nan_K2_v2_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Nan_K2_v2_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Nan_K2_v2 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
```

```

# Must be set to FALSE for pegas implementation of
# permutation testing (for missing data)
method = 'pegas' # Slower but more carefully developed than ade4
)

```

Nangur K=3

`spinosa_Dataset1_Nan` Here I use the DAPC K=3 groupings to see how important that is.

Groupings are Far East, Mid East, and West.

```

# Define individuals as being in the Far East population based on their sampling cluster in the metadata
genlight_spinosa_Dataset1_Nan_K3 <- gl.define.pop(
  genlight_spinosa_Dataset1_Nan,
  ind.list = genlight_spinosa_Dataset1_Nan$ind.names[
    genlight_spinosa_Dataset1_Nan$ind.names %in%
      NanguraSpinosa.dat$INDV[
        NanguraSpinosa.dat$Sample_cluster == 'Nangur_fareast']],
  new = 'Far East')

# Define individuals as being in the Mid East population based on their sampling cluster in the metadata
genlight_spinosa_Dataset1_Nan_K3 <- gl.define.pop(
  genlight_spinosa_Dataset1_Nan_K3,
  ind.list = genlight_spinosa_Dataset1_Nan_K3$ind.names[
    genlight_spinosa_Dataset1_Nan_K3$ind.names %in% NanguraSpinosa.dat$INDV[
      NanguraSpinosa.dat$Sample_cluster == 'Nangur_mideast' &
        NanguraSpinosa.dat$INDV != 'BSU0532']], new = 'Mid East')

# Rename the other population 'West'
genlight_spinosa_Dataset1_Nan_K3 <- gl.rename.pop(
  genlight_spinosa_Dataset1_Nan_K3, old = 'pop1', new = 'West')

# Create strata for the AMOVA using the population information
strata(genlight_spinosa_Dataset1_Nan_K3) <- data.frame(
  'Population' = genlight_spinosa_Dataset1_Nan_K3$pop)

```

Performing the AMOVA using poppr

```

# Perform AMOVA
amova_spinosa_Dataset1_Nan_K3 <- genlight_spinosa_Dataset1_Nan_K3 |>
  poppr.amova(hier = ~Population, # Formula for what strata to analyse
    within = TRUE, # Whether to look within individuals for variation.
    # Must be set to FALSE for pegas implementation of
    # permutation testing (for missing data)
    method = 'ade4'
  )

# Significance test amova using permutation testing ade4 package
amova_spinosa_Dataset1_Nan_K3_sig <- randtest(
  xtest = amova_spinosa_Dataset1_Nan_K3, nrepet = 1000)

# AMOVA results
amova_spinosa_Dataset1_Nan_K3

```

```

# Significance results
amova_spinosa_Dataset1_Nan_K3_sig

# Visualise significance
plot(amova_spinosa_Dataset1_Nan_K3_sig)

# Compare with pegas to double check validity of results
genlight_spinosa_Dataset1_Nan_K3 |>
poppr.amova(hier = ~Population, # Formula for what strata to analyse
  within = TRUE, # Whether to look within individuals for variation.
  #Must be set to FALSE for pegas implementation of
  # permutation testing (for missing data)
  method = 'pegas' # Slower but more carefully developed than ade4
)

```

Private alleles

Private alleles are alleles that are only found in one population. If a population has many private alleles it demonstrates their unique diversity. If populations have many private alleles when compared only to one other population, it demonstrates a lack of connectivity and lots of unique diversity between them. If only one of the populations has many private alleles compared to another population, but not vice versa, it could mean that one of the populations has lost a lot of genetic diversity or that it could indicate uni-directional gene flow. Other results would be needed to piece this together, but importantly by understanding private alleles, we can increase understanding of connectivity and genetic diversity.

I use the finest scale structure K=3 to assess these, because it captures the information present at K=2 also.

Allele Frequency Difference (AFD) is also reported here (Berner et al., 2019). It is similar to Fst but more intuitive.

Species Level

```

gl.report.pa(x = genlight_spinosa_Dataset1_wild_K2, method = 'one2rest')

gl.report.pa(x = genlight_spinosa_Dataset1_wild_FSfilter_K2,
  method = 'pairwise')

```

Oakview

```

gl.report.pa(x = genlight_spinosa_Dataset1_Oak_wild_K3, method = 'one2rest')

```

Nangur

```

gl.report.pa(x = genlight_spinosa_Dataset1_Nan_K3, method = 'one2rest')

```

Fst

```
genlight_spinosa_Dataset1_wild_K2 |>  
  gl.fst.pop(nboots = 100)
```

```
genlight_spinosa_Dataset1_wild_FSfilter_K2 |>  
  gl.fst.pop(nboots = 100)
```

```
genlight_spinosa_Dataset1_Oak_wild_K2 |>  
  gl.fst.pop(nboots = 100)
```

```
genlight_spinosa_Dataset1_Oak_wild_K3 |>  
  gl.fst.pop(nboots = 100)
```

```
genlight_spinosa_Dataset1_Nan_K2 |>  
  gl.fst.pop(nboots = 100)
```

```
genlight_spinosa_Dataset1_Nan_K2_v2 |>  
  gl.fst.pop(nboots = 100)
```

```
genlight_spinosa_Dataset1_Nan_K3 |>  
  gl.fst.pop(nboots = 100)
```

6. Genomic Distance

Connectivity and Structure: Genomic Distance:

The purpose of this analysis is to:

- To use a substitute for kinship appropriate for small, inbred, populations.
- Provide extremely fine-scale (individual basis) information about connectivity and structure.
- To identify the most valuable (i.e., genetically unique) individuals in the dataset.
- To identify the best pairs to breed together in each population.
- To identify whether isolation by distance is the best way to describe how relatedness between individuals declines.

Kinship (not relatedness) is recommended as the best method to determine which populations to cross, which individuals to breed together, and where to put individuals that are being reintroduced (Frankham et al., 2017). Pedigree estimates are often recommended, but they are outperformed by estimates using thousands of SNPs (Frankham et al., 2017). However, an issue with using SNP based methods for kinship is that they are biased by small, inbred samples UNLESS a non-inbred reference population is used to provide allele frequencies (this could be the current population, or a past or future generation of the same population; Wang, 2014).

Fortunately, there is a way around the issue with kinship: by using a measure of genomic distance. Genomic distance is calculated as $1 - \text{identity by state (IBS)}$ across all sites. At each biallelic site IBS is equal to 1 if the genotype is equal, 0.5 if one of the alleles is shared, and 0 if neither of the alleles are shared (Plink manual). Genomic distance has been empirically demonstrated to be useful in managing inbred populations: individuals who were inbred had improved fitness if their mate was genomically distant from them, the

higher the distance the better (Harrison et al., 2019). Coincidentally, pairwise genomic distance is the same measure used to perform the Netview analysis.

An important note. For Oakview, I focus mainly on born-in-the-wild individuals, because the breeding that has occurred in captivity does not reflect that natural processes in Oakview that I am trying to manage and understand.

Prepare dataframes

Here transform the distance matrix into a three column dataframe showing pairwise genomic distances

```
# Give Plink's genomic distance matrix the row names and column names
# in the correct order from netview_id. Note that individuals for
# spinosa_Dataset1 have not been filtered out and there is no HWP based
# filtering.
row.names(mdistspinosa_Dataset1_preclean) <- NanguraSpinosa.id$V1
colnames(mdistspinosa_Dataset1_preclean) <- NanguraSpinosa.id$V1

# Reshape the pairwise matrix into a dataframe with a single comparison per row.
# Note that I extract information from the upper triangle AND I include the
# diagonal so that every individual is compared with itself.
# And then I include the lower triangle so I get indA -> indB AND indB -> indA.
# These comparisons are symmetrical, but it is useful to have both for my calculations afterwards.
distance_spinosa_Dataset1_preclean.a <- melt(mdistspinosa_Dataset1_preclean)[
  melt(upper.tri(mdistspinosa_Dataset1_preclean, diag = TRUE))$value,]
# upper triangle and diagonal

distance_spinosa_Dataset1_preclean.b <- melt(mdistspinosa_Dataset1_preclean)[
  melt(lower.tri(mdistspinosa_Dataset1_preclean, diag = FALSE))$value,] # lower

# Bind the two with rbind.
distance_spinosa_Dataset1_preclean <- rbind(
  distance_spinosa_Dataset1_preclean.a, distance_spinosa_Dataset1_preclean.b)

# Remove the intermediary dataframes
rm(distance_spinosa_Dataset1_preclean.a, distance_spinosa_Dataset1_preclean.b)

# Rename each column to something less generic.
# Herein, DISS refers to genomic distance (dissimilarity)
# DIST refers to geographic distance
colnames(distance_spinosa_Dataset1_preclean) <- c('INDV1', 'INDV2', 'DISS')
```

Now that I have this table. I need to subset it so it just has the comparisons that are relevant to each dataset (so I am making within population comparisons). I can do that using the genlight objects.

```
# Take the subset of comparisons that are only found in the:
# spinosa_Dataset1_Nan dataset
distance_spinosa_Dataset1_Nan <- distance_spinosa_Dataset1_preclean[
  distance_spinosa_Dataset1_preclean$INDV1 %in%
  genlight_spinosa_Dataset1_Nan$ind.names &
  distance_spinosa_Dataset1_preclean$INDV2 %in%
  genlight_spinosa_Dataset1_Nan$ind.names , ]
```

```
# spinosa_Dataset1_Oak
distance_spinosa_Dataset1_Oak <- distance_spinosa_Dataset1_preclean[
  distance_spinosa_Dataset1_preclean$INDV1 %in%
  genlight_spinosa_Dataset1_Oak$ind.names &
  distance_spinosa_Dataset1_preclean$INDV2 %in%
  genlight_spinosa_Dataset1_Oak$ind.names , ]
```

Individual mean pairwise genomic distance

Now I want to calculate individual mean genomic distance. An individual's distance to all other individuals in that population (in that dataset), the average of all individual's pairwise genomic distances including with itself. The higher the mean genomic dissimilarity the more unique the individual is and hence the more valuable it is to contribute the next generation. Similar to the concept of individual mean kinship.

spinosa_Dataset1

```
## For cleanPop datasets...

# Generate a list of individual names for each pop by
# taking only the unique values from the dataframe's first column.
Indv_spinosa_Dataset1_Nan <- unique(distance_spinosa_Dataset1_Nan$INDV1)
Indv_spinosa_Dataset1_Oak <- unique(distance_spinosa_Dataset1_Oak$INDV1)

# Create empty vectors for individual mean genomic distance for each population.
mgd_ind.spinosa_Dataset1_Nan <- NA
mgd_ind.spinosa_Dataset1_Oak <- NA

# A for loop that calculates the mean genomic distance for each individual.
# It does this by averaging across every pairwise genomic distance that
# individual has with every other individual, including with itself.

for (i in 1:length(Indv_spinosa_Dataset1_Nan)) {
  mgd_ind.spinosa_Dataset1_Nan[i] <- mean(distance_spinosa_Dataset1_Nan$DISS[
    distance_spinosa_Dataset1_Nan$INDV1 == Indv_spinosa_Dataset1_Nan[i]])
}

for (i in 1:length(Indv_spinosa_Dataset1_Oak)) {
  mgd_ind.spinosa_Dataset1_Oak[i] <- mean(distance_spinosa_Dataset1_Oak$DISS[
    distance_spinosa_Dataset1_Oak$INDV1 == Indv_spinosa_Dataset1_Oak[i]])
}

# Adding individual names to the file

mgd_ind.spinosa_Dataset1_Nan <- tibble(
  'INDV' = Indv_spinosa_Dataset1_Nan,
  'MEAN.DISS' = mgd_ind.spinosa_Dataset1_Nan)
mgd_ind.spinosa_Dataset1_Oak <- tibble(
  'INDV' = Indv_spinosa_Dataset1_Oak,
  'MEAN.DISS' = mgd_ind.spinosa_Dataset1_Oak)
```

Does geographic distance explain genomic distance?

spinosa_Dataset1_Nan

The genetic distance matrix has already been generated and converted into a dataframe. The geographic distance matrix now needs to be converted in the same way (through matrix melting). I do not run a full statistical model because the effort required is not justified by my need to know the answer. However, a smoothing function is used to generally show the pattern.

Construct a dataframe...

```
# Give the geographic distance matrix row names and
# column names in the correct order.
rownames(distmat_spinosa_Dataset1_Nan) <- NanguraSpinosa.dat$INDV[
  NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_Nan$ind.names]

colnames(distmat_spinosa_Dataset1_Nan) <- NanguraSpinosa.dat$INDV[
  NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_Nan$ind.names]

# Reshape the pairwise matrix into a dataframe with a single comparison per row.
# Note that I extract information from the upper triangle AND I include the
# diagonal so that every individual is compared with itself.
# And then I include the lower triangle so I get indA -> indB AND indB -> indA.
# These comparisons are symmetrical, but it matches other dataframes where I
# needed symmetrical comparisons.
distance_spinosa_Dataset1_Nan.a <- melt(distmat_spinosa_Dataset1_Nan)[
  melt(upper.tri(distmat_spinosa_Dataset1_Nan, diag = TRUE))$value,]
# upper triangle and diagonal

distance_spinosa_Dataset1_Nan.b <- melt(distmat_spinosa_Dataset1_Nan)[
  melt(lower.tri(distmat_spinosa_Dataset1_Nan, diag = FALSE))$value,] # lower

# Bind the two with rbind.
distance_spinosa_Dataset1_Nan <- rbind(distance_spinosa_Dataset1_Nan.a,
                                       distance_spinosa_Dataset1_Nan.b)

# Remove the intermediary dataframes
rm(distance_spinosa_Dataset1_Nan.a, distance_spinosa_Dataset1_Nan.b)

# Rename each column to something less generic.
colnames(distance_spinosa_Dataset1_Nan) <- c('INDV1', 'INDV2', 'DIST')
```

Merge dataframes...

```
# Merge the two dataframes geographic distances and pairwise kinship
MGDXDIST_spinosa_Dataset1_Nan <- merge(x = distance_spinosa_Dataset1_Nan,
                                       y = distance_spinosa_Dataset1_Nan,
                                       by.x = c('INDV1', 'INDV2'),
                                       by.y = c('INDV1', 'INDV2'))
```

Plot...

```

# Now plot the relationship, a linear model is not
# really appropriate for this pattern
ggplot(MGDXDIST.spinosa_Dataset1_Nan,
       aes(x = DIST, y = DISS)) +
  geom_point(colour = 'forestgreen') +
  theme_classic() +
  xlab('Pairwise distance (m)') +
  ylab('Pairwise genomic distance')

```

spinosa_Dataset1_Oak

The genetic distance matrix has already been generated and converted into a dataframe. The geographic distance matrix now needs to be converted in the same way (through matrix melting). I do not run a full statistical model because the effort required is not justified by my need to know the answer. However, a smoothing function is used to generally show the pattern.

Construct a dataframe...

```

# Give the geographic distance matrix row names and
# column names in the correct order.
rownames(distmat_spinosa_Dataset1_Oak) <- NanguraSpinosa.dat$INDV[
  NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_Oak_wild$ind.names]

colnames(distmat_spinosa_Dataset1_Oak) <- NanguraSpinosa.dat$INDV[
  NanguraSpinosa.dat$INDV %in% genlight_spinosa_Dataset1_Oak_wild$ind.names]

# Reshape the pairwise matrix into a dataframe with a single comparison per row.
# Note that I extract information from the upper triangle AND
# I include the diagonal so that every individual is compared with itself.
# And then I include the lower triangle so I get indA -> indB AND indB -> indA.
# These comparisons are symmetrical,
# but it matches other dataframes where I needed symmetrical comparisons.
distance_spinosa_Dataset1_Oak.a <- melt(distmat_spinosa_Dataset1_Oak)[
  melt(upper.tri(distmat_spinosa_Dataset1_Oak, diag = TRUE))$value,]
# upper triangle and diagonal

distance_spinosa_Dataset1_Oak.b <- melt(distmat_spinosa_Dataset1_Oak)[
  melt(lower.tri(distmat_spinosa_Dataset1_Oak, diag = FALSE))$value,] # lower

# Bind the two with rbind.
distance_spinosa_Dataset1_Oak <- rbind(distance_spinosa_Dataset1_Oak.a,
                                       distance_spinosa_Dataset1_Oak.b)

# Remove the intermediary dataframes
rm(distance_spinosa_Dataset1_Oak.a, distance_spinosa_Dataset1_Oak.b)

# Rename each column to something less generic.
colnames(distance_spinosa_Dataset1_Oak) <- c('INDV1', 'INDV2', 'DIST')

```

Merge dataframes...

```
# Merge the two dataframes geographic distances and pairwise kinship
MGDXDIST.spinosa_Dataset1_Oak <- merge(x = distance_spinosa_Dataset1_Oak,
                                       y = distance_spinosa_Dataset1_Oak,
                                       by.x = c('INDV1', 'INDV2'),
                                       by.y = c('INDV1', 'INDV2'))
```

Plot...

```
# Now plot the relationship, a linear model is not
# really appropriate for this pattern
ggplot(MGDXDIST.spinosa_Dataset1_Oak,
       aes(x = DIST, y = DISS)) +
  geom_point(colour = 'royalblue') +
  theme_classic() +
  xlab('Pairwise distance (m)') +
  ylab('Pairwise genomic distance') +
  scale_x_continuous(n.breaks = 16)
```

Combine the plots

```
# Add a column that names the population the comparison comes from
MGDXDIST.spinosa_Dataset1_Nan$POP <- 'Nangur'

MGDXDIST.spinosa_Dataset1_Oak$POP <- 'Oakview'

# Bind the two dataframes together
MGDXDIST.joined <- rbind(MGDXDIST.spinosa_Dataset1_Nan,
                         MGDXDIST.spinosa_Dataset1_Oak)
```

```
MGDXDIST.joined[] |>
ggplot(
  aes(x = DIST, y = DISS, colour = POP)) +
  geom_point(alpha = 0.5, size = 2) +
  scale_colour_manual(values = c('forestgreen', 'royalblue')) +
  labs(colour = 'Population') +
  theme_classic() +
  xlab('Pairwise geographic distance (m)') +
  ylab('Pairwise genomic distance') +

  # Set where the x axis breaks and labels occur
  scale_x_continuous(breaks = c(0, 500, seq(from = 1000, to = 6000,
                                           by = 1000))) +

  # Add a locally estimated scatterplot smoothing term
  # that draws a line through the data, colour the 95% confidence interval,
  # and use a span (the degree of smoothing of 0.75 [DEFAULT])
  geom_smooth(aes(group = POP), colour = 'black', fill = 'grey20',
              method = 'loess') +

  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
  geom_point(aes(x = 0, y = 0), colour = 'black', size = 2)
```

Average genomic distance between and within populations

This will be useful to compare and understand scale. It replicates what is recommended for kinship.

Without full-sib filter

```
# Take the spinosa_Dataset1_wild subset of individual comparisons
distance_spinosa_Dataset1 <- distance_spinosa_Dataset1_preclean[
  distance_spinosa_Dataset1_preclean$INDV1 %in%
  genlight_spinosa_Dataset1_wild$ind.names &
  distance_spinosa_Dataset1_preclean$INDV2 %in%
  genlight_spinosa_Dataset1_wild$ind.names , ]

# Name the populations of Individual 1...
# ...if they are from Nangur
distance_spinosa_Dataset1$POP1[distance_spinosa_Dataset1$INDV1 %in%
  genlight_spinosa_Dataset1_Nan$ind.names] <-
  'Nangur'

# ...if they are from Oakview
distance_spinosa_Dataset1$POP1[
  distance_spinosa_Dataset1$INDV1 %in%
  genlight_spinosa_Dataset1_Oak_wild$ind.names] <- 'Oakview'

# Name the populations of Individual 2...
# ...if they are from Nangur
distance_spinosa_Dataset1$POP2[
  distance_spinosa_Dataset1$INDV2 %in%
  genlight_spinosa_Dataset1_Nan$ind.names] <-
  'Nangur'

# ...if they are from Oakview
distance_spinosa_Dataset1$POP2[
  distance_spinosa_Dataset1$INDV2 %in%
  genlight_spinosa_Dataset1_Oak_wild$ind.names] <- 'Oakview'

# Calculate the between population mean pairwise genomic distance
paste0('between population mean: ',
  mean(distance_spinosa_Dataset1$DISS[
    distance_spinosa_Dataset1$POP1 == 'Nangur' &
    distance_spinosa_Dataset1$POP2 == 'Oakview'])))

# Calculate the within population mean individual pairwise
# genomic distance for Nangur
paste0('within population mean (Nangur): ',
  mean(distance_spinosa_Dataset1$DISS[
    distance_spinosa_Dataset1$POP1 == 'Nangur' &
```

```

distance_spinosa_Dataset1$POP2 == 'Nangur']))

# Calculate the standard error for the same
sd(distance_spinosa_Dataset1$DISS[
  distance_spinosa_Dataset1$POP1 == 'Nangur' &
  distance_spinosa_Dataset1$POP2 == 'Nangur'])/sqrt(length(distance_spinosa_Dataset1$DISS[
  distance_spinosa_Dataset1$POP1 == 'Nangur' &
  distance_spinosa_Dataset1$POP2 == 'Nangur']))

# Calculate the within population mean individual pairwise
# genomic distance for Oakview
paste0('within population mean (Oakview): ',
  mean(distance_spinosa_Dataset1$DISS[
    distance_spinosa_Dataset1$POP1 == 'Oakview'
    & distance_spinosa_Dataset1$POP2 == 'Oakview']))

# Calculate the standard error for the same
sd(distance_spinosa_Dataset1$DISS[
  distance_spinosa_Dataset1$POP1 == 'Oakview'
  & distance_spinosa_Dataset1$POP2 == 'Oakview'])/sqrt(length(distance_spinosa_Dataset1$DISS[
  distance_spinosa_Dataset1$POP1 == 'Oakview'
  & distance_spinosa_Dataset1$POP2 == 'Oakview']))

```

With full-sib filter

```

# Take the spinosa_Dataset1_wild subset of individual comparisons
distance_spinosa_Dataset1_FSfilter <- distance_spinosa_Dataset1_preclean[
  distance_spinosa_Dataset1_preclean$INDV1 %in%
  genlight_spinosa_Dataset1_wild_FSfilter$ind.names &
  distance_spinosa_Dataset1_preclean$INDV2 %in%
  genlight_spinosa_Dataset1_wild_FSfilter$ind.names , ]

# Name the populations of Individual 1...
# ...if they are from Nangur
distance_spinosa_Dataset1_FSfilter$POP1[
  distance_spinosa_Dataset1_FSfilter$INDV1 %in%
  genlight_spinosa_Dataset1_Nan$ind.names] <- 'Nangur'

# ...if they are from Oakview
distance_spinosa_Dataset1_FSfilter$POP1[
  distance_spinosa_Dataset1_FSfilter$INDV1 %in%
  genlight_spinosa_Dataset1_Oak_wild$ind.names] <- 'Oakview'

# Name the populations of Individual 2...
# ...if they are from Nangur
distance_spinosa_Dataset1_FSfilter$POP2[
  distance_spinosa_Dataset1_FSfilter$INDV2 %in%
  genlight_spinosa_Dataset1_Nan$ind.names] <- 'Nangur'

# ...if they are from Oakview
distance_spinosa_Dataset1_FSfilter$POP2[
  distance_spinosa_Dataset1_FSfilter$INDV2 %in%

```

```

genlight_spinosa_Dataset1_Oak_wild$ind.names] <- 'Oakview'

# Calculate the between population mean pairwise genomic distance
paste0('between population mean: ',
       mean(distance_spinosa_Dataset1_FSfilter$DISS[
         distance_spinosa_Dataset1_FSfilter$POP1 == 'Nangur' &
         distance_spinosa_Dataset1_FSfilter$POP2 == 'Oakview'])))

# Calculate the within population mean individual pairwise
# genomic distance for Nangur
paste0('within population mean (Nangur): ',
       mean(distance_spinosa_Dataset1_FSfilter$DISS[
         distance_spinosa_Dataset1_FSfilter$POP1 == 'Nangur' &
         distance_spinosa_Dataset1_FSfilter$POP2 == 'Nangur'])))
# Calculate standard error for the same
sd(distance_spinosa_Dataset1_FSfilter$DISS[
  distance_spinosa_Dataset1_FSfilter$POP1 == 'Nangur' &
  distance_spinosa_Dataset1_FSfilter$POP2 == 'Nangur'])/sqrt(length(distance_spinosa_Dataset1_
  distance_spinosa_Dataset1_FSfilter$POP1 == 'Nangur' &
  distance_spinosa_Dataset1_FSfilter$POP2 == 'Nangur'])))

# Calculate the within population mean individual pairwise genomic distance for
# Oakview
paste0('within population mean (Oakview): ',
       mean(distance_spinosa_Dataset1_FSfilter$DISS[
         distance_spinosa_Dataset1_FSfilter$POP1 == 'Oakview' &
         distance_spinosa_Dataset1_FSfilter$POP2 == 'Oakview'])))

# Calculate standard error for the same
sd(distance_spinosa_Dataset1_FSfilter$DISS[
  distance_spinosa_Dataset1_FSfilter$POP1 == 'Oakview' &
  distance_spinosa_Dataset1_FSfilter$POP2 == 'Oakview'])/sqrt(length(distance_spinosa_Dataset1_
  distance_spinosa_Dataset1_FSfilter$POP1 == 'Oakview' &
  distance_spinosa_Dataset1_FSfilter$POP2 == 'Oakview'])))

```

Average pairwise genomic distance between full sibs and halfsibs.

As a reference point to other observations, I want to find the average pairwise genomic distance between full sibs and halfsibs, separately for each population.

Fullsibs

```

# Read in the list of full-sib pairs
Nangur.PairwiseFullSibDyad <- read.csv(
  file = '../data/other_files/spinosa_Dataset1_Nan.PairwiseFullSibDyad')

# Create new vectors for the genomic distance and geographic distance
Nangur.PairwiseFullSibDyad$DISS <- NA

Nangur.PairwiseFullSibDyad$DIST <- NA

```

```

# A for loop.
# For each row (pair of full sibs) in the list of full sibs,
# Find the corresponding row in the whole population distance table.
# Then record the distance and distance for that pair.
for (i in 1:nrow(Nangur.PairwiseFullSibDyad)){

  Nangur.PairwiseFullSibDyad$DISS[i] <- MGDXDIST.spinosa_Dataset1_Nan$DISS[
    (MGDXDIST.spinosa_Dataset1_Nan$INDV1 ==
     Nangur.PairwiseFullSibDyad$OffspringID1[i]) &
    (MGDXDIST.spinosa_Dataset1_Nan$INDV2 ==
     Nangur.PairwiseFullSibDyad$OffspringID2[i])]

  Nangur.PairwiseFullSibDyad$DIST[i] <- MGDXDIST.spinosa_Dataset1_Nan$DIST[
    (MGDXDIST.spinosa_Dataset1_Nan$INDV1 ==
     Nangur.PairwiseFullSibDyad$OffspringID1[i]) &
    (MGDXDIST.spinosa_Dataset1_Nan$INDV2 ==
     Nangur.PairwiseFullSibDyad$OffspringID2[i])]

}

# Calculate summary statistics
summary(Nangur.PairwiseFullSibDyad$DISS)

summary(Nangur.PairwiseFullSibDyad$DIST)

summary(Nangur.PairwiseFullSibDyad$DIST[Nangur.PairwiseFullSibDyad$DIST != 0])

```

Nangur

Oakview Note that this is wild only

```

# Read in the list of full-sib pairs
Oakview.PairwiseFullSibDyad <- read.csv(
  file = '../data/other_files/spinosa_Dataset1_Oak.PairwiseFullSibDyad')

# Create new vectors for the genomic distance and geographic distance
Oakview.PairwiseFullSibDyad$DISS <- NA

Oakview.PairwiseFullSibDyad$DIST <- NA

# Remove individuals that are not born in the wild.
Oakview.PairwiseFullSibDyad.wild <- Oakview.PairwiseFullSibDyad[
  (Oakview.PairwiseFullSibDyad$OffspringID1 %in%
   MGDXDIST.spinosa_Dataset1_Oak$INDV1),]

Oakview.PairwiseFullSibDyad.wild <- Oakview.PairwiseFullSibDyad[
  (Oakview.PairwiseFullSibDyad$OffspringID2 %in%
   MGDXDIST.spinosa_Dataset1_Oak$INDV1),]

# A for loop.
# For each row (pair of full sibs) in the list of full sibs,
# Find the corresponding row in the whole population distance table.

```

```

# Then record the distance and distance for that pair.
for (i in 1:nrow(Oakview.PairwiseFullSibDyad.wild)){

  Oakview.PairwiseFullSibDyad.wild$DISS[i] <-
    MGDXDIST.spinosa_Dataset1_Oak$DISS[(
      MGDXDIST.spinosa_Dataset1_Oak$INDV1 ==
      Oakview.PairwiseFullSibDyad.wild$OffspringID1[i]) &
      (MGDXDIST.spinosa_Dataset1_Oak$INDV2 ==
      Oakview.PairwiseFullSibDyad.wild$OffspringID2[i])]

  Oakview.PairwiseFullSibDyad.wild$DIST[i] <-
    MGDXDIST.spinosa_Dataset1_Oak$DIST[(
      MGDXDIST.spinosa_Dataset1_Oak$INDV1 ==
      Oakview.PairwiseFullSibDyad.wild$OffspringID1[i]) &
      (MGDXDIST.spinosa_Dataset1_Oak$INDV2 ==
      Oakview.PairwiseFullSibDyad.wild$OffspringID2[i])]

}

# Calculate summary statistics
summary(Oakview.PairwiseFullSibDyad.wild$DISS)

summary(Oakview.PairwiseFullSibDyad.wild$DIST)

# Not found in same burrow
summary(Oakview.PairwiseFullSibDyad.wild$DIST[
  Oakview.PairwiseFullSibDyad.wild$DIST != 0])

```

Half-sibs

```

# Read in the list of full-sib pairs
Nangur.PairwiseHalfSibDyad <- read.csv(
  file = '../data/other_files/spinosa_Dataset1_Nan.PairwiseHalfSibDyad')

# Create new vectors for the genomic distance and geographic distance
Nangur.PairwiseHalfSibDyad$DISS <- NA

Nangur.PairwiseHalfSibDyad$DIST <- NA

# Remove individuals that are not born in the wild.
Nangur.PairwiseHalfSibDyad.wild <- Nangur.PairwiseHalfSibDyad[(
  Nangur.PairwiseHalfSibDyad$OffspringID1 %in%
  MGDXDIST.spinosa_Dataset1_Nan$INDV1),]

Nangur.PairwiseHalfSibDyad.wild <- Nangur.PairwiseHalfSibDyad[(
  Nangur.PairwiseHalfSibDyad$OffspringID2 %in%
  MGDXDIST.spinosa_Dataset1_Nan$INDV1),]

# A for loop.
# For each row (pair of Half sibs) in the list of Half sibs,
# Find the corresponding row in the whole population distance table.

```

```

# Then record the distance and distance for that pair.
for (i in 1:nrow(Nangur.PairwiseHalfSibDyad.wild)){

  Nangur.PairwiseHalfSibDyad.wild$DISS[i] <-
    MGDXDIST.spinosa_Dataset1_Nan$DISS[(
      MGDXDIST.spinosa_Dataset1_Nan$INDV1 ==
        Nangur.PairwiseHalfSibDyad.wild$OffspringID1[i]) &
      (MGDXDIST.spinosa_Dataset1_Nan$INDV2 ==
        Nangur.PairwiseHalfSibDyad.wild$OffspringID2[i])]

  Nangur.PairwiseHalfSibDyad.wild$DIST[i] <-
    MGDXDIST.spinosa_Dataset1_Nan$DIST[(
      MGDXDIST.spinosa_Dataset1_Nan$INDV1 ==
        Nangur.PairwiseHalfSibDyad.wild$OffspringID1[i]) &
      (MGDXDIST.spinosa_Dataset1_Nan$INDV2 ==
        Nangur.PairwiseHalfSibDyad.wild$OffspringID2[i])]

}

# Calculate summary statistics
summary(Nangur.PairwiseHalfSibDyad.wild$DISS)

summary(Nangur.PairwiseHalfSibDyad.wild$DIST)

```

Nangur

```

# Read in the list of full-sib pairs
Oakview.PairwiseHalfSibDyad <- read.csv(
  file = '../data/other_files/spinosa_Dataset1_Oak.PairwiseHalfSibDyad')

# Create new vectors for the genomic distance and geographic distance
Oakview.PairwiseHalfSibDyad$DISS <- NA

Oakview.PairwiseHalfSibDyad$DIST <- NA

# Remove individuals that are not born in the wild.
Oakview.PairwiseHalfSibDyad.wild <- Oakview.PairwiseHalfSibDyad[(
  Oakview.PairwiseHalfSibDyad$OffspringID1 %in%
    MGDXDIST.spinosa_Dataset1_Oak$INDV1),]

Oakview.PairwiseHalfSibDyad.wild <- Oakview.PairwiseHalfSibDyad[(
  Oakview.PairwiseHalfSibDyad$OffspringID2 %in%
    MGDXDIST.spinosa_Dataset1_Oak$INDV1),]

# A for loop.
# For each row (pair of Half sibs) in the list of Half sibs,
# Find the corresponding row in the whole population distance table.
# Then record the distance and distance for that pair.
for (i in 1:nrow(Oakview.PairwiseHalfSibDyad.wild)){

```

```

Oakview.PairwiseHalfSibDyad.wild$DISS[i] <-
  MGDXDIST.spinosa_Dataset1_Oak$DISS[(
    MGDXDIST.spinosa_Dataset1_Oak$INDV1 ==
    Oakview.PairwiseHalfSibDyad.wild$OffspringID1[i]) &
    (MGDXDIST.spinosa_Dataset1_Oak$INDV2 ==
    Oakview.PairwiseHalfSibDyad.wild$OffspringID2[i])]

Oakview.PairwiseHalfSibDyad.wild$DIST[i] <-
  MGDXDIST.spinosa_Dataset1_Oak$DIST[(
    MGDXDIST.spinosa_Dataset1_Oak$INDV1 ==
    Oakview.PairwiseHalfSibDyad.wild$OffspringID1[i]) &
    (MGDXDIST.spinosa_Dataset1_Oak$INDV2 ==
    Oakview.PairwiseHalfSibDyad.wild$OffspringID2[i])]
}

# Calculate summary statistics
summary(Oakview.PairwiseHalfSibDyad.wild$DISS)

summary(Oakview.PairwiseHalfSibDyad.wild$DIST)

```

Oakview

The genomic distance plot with full and halfsibs marked

Fullsibs marked in yellow Halfsibs marked in pink

```

MGDXDIST.joined[] |>
ggplot(
  aes(x = DIST, y = DISS, colour = POP)) +
  geom_point(alpha = 0.5, size = 2) +
  scale_colour_manual(values = c('yellow', '#FF647F',
                                'forestgreen', '#0000ff')) +
  labs(colour = 'Identity of pair') +
  theme_classic() +
  xlab('Pairwise geographic distance (m)') +
  ylab('Pairwise genomic distance') +

  # Set where the x axis breaks and labels occur
  scale_x_continuous(breaks = c(0, 500, seq(from = 1000, to = 6000,
                                             by = 1000))) +

  # Add a locally estimated scatterplot smoothing term
  # that draws a line through the data, colour the 95% confidence interval,
  # and use a span (the degree of smoothing of 0.75 [DEFAULT])
  geom_smooth(aes(group = POP), colour = 'black', fill = 'grey20',
              method = 'loess') +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2),
    legend.key = element_rect(fill = 'grey90')) +

```

```

# Annotate the origin
geom_point(aes(x = 0, y = 0), colour = 'black', size = 2) +

# Annotate on all individuals who are half sibs...

geom_point(data = Oakview.PairwiseHalfSibDyad.wild,
  aes(x = Oakview.PairwiseHalfSibDyad.wild$DIST,
    y = Oakview.PairwiseHalfSibDyad.wild$DISS,
    colour = '2nd-degree relationship'), size = 0.6) +

geom_point(data = Nangur.PairwiseHalfSibDyad.wild,
  aes(x = Nangur.PairwiseHalfSibDyad.wild$DIST,
    y = Nangur.PairwiseHalfSibDyad.wild$DISS,
    colour = '2nd-degree relationship'), size = 0.6) +

# ...and full-sibs...
geom_point(data = Oakview.PairwiseFullSibDyad.wild,
  aes(x = Oakview.PairwiseFullSibDyad.wild$DIST,
    y = Oakview.PairwiseFullSibDyad.wild$DISS,
    colour = '1st-degree relationship'), size = 0.6) +

geom_point(data = Nangur.PairwiseFullSibDyad,
  aes(x = Nangur.PairwiseFullSibDyad$DIST,
    y = Nangur.PairwiseFullSibDyad$DISS,
    colour = '1st-degree relationship'), size = 0.6)

```

7. Neighbourhood Radius Size

Connectivity and structure: Neighbourhoods:

The purpose of this analysis is to: * Identify the radius of a genetic ‘neighbourhood’ for *Nangura spinosa* using the software sGD. * Assist in understanding the processes affecting the internal structure of *Nangura spinosa* populations. * Inform information about *Nangura spinosa* dispersal. * Determine the scale at which certain analyses should take place (e.g., effective population size)

I use the method suggested by Neel et al. (2013), and Shirk and Cushman (2014), and the sGD package itself, to roughly estimate the radius of Wright’s neighbourhood. This method involves grouping individuals into putative neighbourhoods over several possible neighbourhood radii. Then calculating F_{is} for each neighbourhood for each radius. I use trial and error until I find the radius that produces an F_{is} distribution is centred on zero. The function `infer2sigma` from the sGD package does this for me. This neighbourhood radius is an estimate of twice the standard deviation of the parent-offspring dispersal distance. The neighbourhood itself is estimated as a circle, representing an area that could be considered panmictic, in other words, an area representing where an individual’s parents most likely came from. An individuals at the centre of the neighbourhood are expected to have 86.5% of their parents within the neighbourhood.

Infering Neighbourhood radius

genind_spinosa_Dataset1_Nan

```
# Set which dataset, latlongs, and distance matrix I'll be using.

genind <- genind_spinosa_Dataset1_Nan
latlong <- latlong_spinosa_Dataset1_Nan
distmat <- distmat_spinosa_Dataset1_Nan


# The function infer2sigma was designed to run the sGD function several times (calculating Fis) and the

sGD.400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.700 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 700, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.800 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 800, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```

)

sGD.900 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 900, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
  #NHmat_ans = T,
  #genout_ans = T,
  #file_name = 'sGD.900'
)

sGD.1000 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1000, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1100 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1100, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1200 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1200, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1300 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1300, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

```

```
sGD.1500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)
```

```
sGD.1600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)
```

The sGD function is malfunctioning and does not give me Fis values, so I need to calculate them myself.

```
sGD.400$FIS <- 1 - sGD.400$Ho / sGD.400$Hs
sGD.500$FIS <- 1 - sGD.500$Ho / sGD.500$Hs
sGD.600$FIS <- 1 - sGD.600$Ho / sGD.600$Hs
sGD.700$FIS <- 1 - sGD.700$Ho / sGD.700$Hs
sGD.800$FIS <- 1 - sGD.800$Ho / sGD.800$Hs
sGD.900$FIS <- 1 - sGD.900$Ho / sGD.900$Hs
sGD.1000$FIS <- 1 - sGD.1000$Ho / sGD.1000$Hs
sGD.1100$FIS <- 1 - sGD.1100$Ho / sGD.1100$Hs
sGD.1200$FIS <- 1 - sGD.1200$Ho / sGD.1200$Hs
sGD.1300$FIS <- 1 - sGD.1300$Ho / sGD.1300$Hs
sGD.1400$FIS <- 1 - sGD.1400$Ho / sGD.1400$Hs
sGD.1500$FIS <- 1 - sGD.1500$Ho / sGD.1500$Hs
sGD.1600$FIS <- 1 - sGD.1600$Ho / sGD.1600$Hs
```

Label the neighbourhoods by the central individuals they are centred on.

```
sGD.400$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.500$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.600$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.700$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.800$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.900$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1000$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1100$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1200$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1300$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1400$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1500$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1600$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
```

```
sGD_FIS <- c(sGD.400$FIS, sGD.500$FIS, sGD.600$FIS, sGD.700$FIS, sGD.800$FIS, sGD.900$FIS, sGD.1000$FIS,
sGD_radii <- rep(c(400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600), each = nrow
```

```

sgd_2sigma.spinosa_Dataset1_Nan <- tibble(sgd_radrii, sgd_FIS)

ggplot(sgd_2sigma.spinosa_Dataset1_Nan,
       aes(y = sgd_FIS, x = as.factor(sgd_radrii), fill = as.factor(sgd_radrii))) +
  geom_boxplot(fill = 'grey')+
  geom_point()+
  geom_hline(yintercept = 0, colour = 'red')+
  guides(fill = FALSE) +
  theme_classic()+
  xlab('Neighbourhood Radius') +
  ylab(bquote( F[IS])) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2))

ggplot(sgd_2sigma.spinosa_Dataset1_Nan, aes(x = as.factor(sgd_radrii), y = sgd_FIS)) +
  geom_violin(fill = 'grey')+
  geom_point()+
  geom_hline(yintercept = 0, colour = 'red')+
  guides(fill = FALSE) +
  theme_classic()+
  xlab('Neighbourhood Radius') +
  ylab(bquote( F[IS])) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2))

summary(sgd.400$FIS)
summary(sgd.500$FIS)
summary(sgd.600$FIS)
summary(sgd.700$FIS)
summary(sgd.800$FIS)
summary(sgd.900$FIS)
summary(sgd.1000$FIS)
summary(sgd.1100$FIS)
summary(sgd.1200$FIS)
summary(sgd.1300$FIS)
summary(sgd.1400$FIS)
summary(sgd.1500$FIS)
summary(sgd.1600$FIS)

```

genind_spinosa_Dataset1_Oak_wild

```

# Set which dataset, latlongs, and distance matrix I'll be using.

genind <- genind_spinosa_Dataset1_Oak_wild
latlong <- latlong_spinosa_Dataset1_Oak

```

```
distmat <- distmat_spinosa_Dataset1_Oak
```

The function infer2sigma was designed to run the sGD function several times (calculating Fis) and the

```
sGD.400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.700 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 700, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.800 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 800, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.900 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 900, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```

    #NHmat_ans = T,
    #genout_ans = T,
    #file_name = 'sGD.900'
  )

sGD.1000 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1000, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1100 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1100, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1200 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1200, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1300 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1300, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

```

```

sGD.1600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)

# The sGD function is malfunctioning and does not give me Fis values, so I need to calculate them myself.

sGD.400$FIS <- 1 - sGD.400$Ho / sGD.400$Hs
sGD.500$FIS <- 1 - sGD.500$Ho / sGD.500$Hs
sGD.600$FIS <- 1 - sGD.600$Ho / sGD.600$Hs
sGD.700$FIS <- 1 - sGD.700$Ho / sGD.700$Hs
sGD.800$FIS <- 1 - sGD.800$Ho / sGD.800$Hs
sGD.900$FIS <- 1 - sGD.900$Ho / sGD.900$Hs
sGD.1000$FIS <- 1 - sGD.1000$Ho / sGD.1000$Hs
sGD.1100$FIS <- 1 - sGD.1100$Ho / sGD.1100$Hs
sGD.1200$FIS <- 1 - sGD.1200$Ho / sGD.1200$Hs
sGD.1300$FIS <- 1 - sGD.1300$Ho / sGD.1300$Hs
sGD.1400$FIS <- 1 - sGD.1400$Ho / sGD.1400$Hs
sGD.1500$FIS <- 1 - sGD.1500$Ho / sGD.1500$Hs
sGD.1600$FIS <- 1 - sGD.1600$Ho / sGD.1600$Hs

# Label the neighbourhoods by the central individuals they are centred on.
sGD.400$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.500$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.600$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.700$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.800$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.900$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1000$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1100$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1200$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1300$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1400$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1500$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV
sGD.1600$NH_ID <- latlong_spinosa_Dataset1_Oak$INDV

sGD_FIS <- c(sGD.400$FIS, sGD.500$FIS, sGD.600$FIS, sGD.700$FIS, sGD.800$FIS, sGD.900$FIS, sGD.1000$FIS,
sGD.1100$FIS, sGD.1200$FIS, sGD.1300$FIS, sGD.1400$FIS, sGD.1500$FIS, sGD.1600$FIS)
sGD_radii <- rep(c(400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600), each = nrow(sGD_FIS))
sGD_2sigma.spinosa_Dataset1_Oak_wild <- tibble(sGD_radii, sGD_FIS)

ggplot(sGD_2sigma.spinosa_Dataset1_Oak_wild,
  aes(y = sGD_FIS, x = as.factor(sGD_radii), fill = as.factor(sGD_radii))) +
  geom_boxplot(fill = 'grey') +
  geom_point() +
  geom_hline(yintercept = 0, colour = 'red') +
  guides(fill = FALSE) +
  theme_classic() +
  xlab('Neighbourhood Radius') +

```

```

ylab(bquote( F[IS])) +
theme(
  # Make axis titles bold
  axis.title = element_text(face = 2))

ggplot(sGD_2sigma.spinosa_Dataset1_Oak_wild, aes(x = as.factor(sGD_radii), y = sGD_FIS)) +
  geom_violin(fill = 'grey')+
  geom_point()+
  geom_hline(yintercept = 0, colour = 'red')+
  guides(fill = FALSE) +
  theme_classic()+
  xlab('Neighbourhood Radius') +
  ylab(bquote( F[IS])) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2))

summary(sGD.400$FIS)
summary(sGD.500$FIS)
summary(sGD.600$FIS)
summary(sGD.700$FIS)
summary(sGD.800$FIS)
summary(sGD.900$FIS)
summary(sGD.1000$FIS)
summary(sGD.1100$FIS)
summary(sGD.1200$FIS)
summary(sGD.1300$FIS)
summary(sGD.1400$FIS)
summary(sGD.1500$FIS)
summary(sGD.1600$FIS)

```

genind_spinosa_Dataset2_wild

```

# Set which dataset, latlongs, and distance matrix I'll be using.

genind <- genind_spinosa_Dataset2_wild
latlong <- latlong_spinosa_Dataset1
distmat <- distmat_spinosa_Dataset1

# The function infer2sigma was designed to run the sGD function several times (calculating Fis) and the

sGD.400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate

```

```

)

sGD.500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.700 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 700, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.800 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 800, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.900 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 900, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
  #NHmat_ans = T,
  #genout_ans = T,
  #file_name = 'sGD.900'
)

sGD.1000 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1000, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

```

```

sGD.1100 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1100, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)

# The sGD function is malfunctioning and does not give me Fis values, so I need to calculate them myself.

sGD.400$FIS <- 1 - sGD.400$Ho / sGD.400$Hs
sGD.500$FIS <- 1 - sGD.500$Ho / sGD.500$Hs
sGD.600$FIS <- 1 - sGD.600$Ho / sGD.600$Hs
sGD.700$FIS <- 1 - sGD.700$Ho / sGD.700$Hs
sGD.800$FIS <- 1 - sGD.800$Ho / sGD.800$Hs
sGD.900$FIS <- 1 - sGD.900$Ho / sGD.900$Hs
sGD.1000$FIS <- 1 - sGD.1000$Ho / sGD.1000$Hs
sGD.1100$FIS <- 1 - sGD.1100$Ho / sGD.1100$Hs

# Label the neighbourhoods by the central individuals they are centred on.
sGD.400$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.500$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.600$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.700$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.800$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.900$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1000$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV
sGD.1100$NH_ID <- latlong_spinosa_Dataset1_Nan$INDV

sGD_FIS <- c(sGD.400$FIS, sGD.500$FIS, sGD.600$FIS, sGD.700$FIS, sGD.800$FIS, sGD.900$FIS, sGD.1000$FIS)
sGD_radII <- rep(c(400, 500, 600, 700, 800, 900, 1000, 1100), each = nrow(sGD.400))
sGD_2sigma <- tibble(sGD_radII, sGD_FIS)

ggplot(sGD_2sigma,
  aes(y = sGD_FIS, x = as.factor(sGD_radII), fill = as.factor(sGD_radII))) +
  geom_boxplot() +
  geom_hline(yintercept = 0, col = 'tomato') +
  theme_classic()

ggplot(sGD_2sigma, aes(x = as.factor(sGD_radII), y = sGD_FIS, fill = as.factor(sGD_radII))) +
  geom_violin() +
  geom_point() +
  geom_hline(yintercept = 0, colour = 'red')

summary(sGD.400$FIS)
summary(sGD.500$FIS)
summary(sGD.600$FIS)
summary(sGD.700$FIS)
summary(sGD.800$FIS)
summary(sGD.900$FIS)
summary(sGD.1000$FIS)

```

```
summary(sGD.1100$FIS)
```

genind_spinosa_Dataset1_Nan_FSfilter

```
# Set which dataset, latlongs, and distance matrix I'll be using.
```

```
genind <- genind_spinosa_Dataset1_Nan_FSfilter  
latlong <- latlong_spinosa_Dataset1_Nan_FSfilter  
distmat <- distmat_spinosa_Dataset1_Nan_FSfilter
```

```
sGD.400 <- sGD(genind, # genetic data  
  xy = latlong, # geographic coordinates  
  dist.mat = distmat, # geographic distance matrix  
  NH_radius = 400, # radius of neighbourhood (chosen a priori) to try  
  min_N = 5, # Minimum sample size to calculate metrics  
  metrics = c('GD'), # CWhat metrics to calculate  
)
```

```
sGD.500 <- sGD(genind, # genetic data  
  xy = latlong, # geographic coordinates  
  dist.mat = distmat, # geographic distance matrix  
  NH_radius = 500, # radius of neighbourhood (chosen a priori) to try  
  min_N = 5, # Minimum sample size to calculate metrics  
  metrics = c('GD'), # CWhat metrics to calculate  
)
```

```
sGD.600 <- sGD(genind, # genetic data  
  xy = latlong, # geographic coordinates  
  dist.mat = distmat, # geographic distance matrix  
  NH_radius = 600, # radius of neighbourhood (chosen a priori) to try  
  min_N = 5, # Minimum sample size to calculate metrics  
  metrics = c('GD'), # CWhat metrics to calculate  
)
```

```
sGD.700 <- sGD(genind, # genetic data  
  xy = latlong, # geographic coordinates  
  dist.mat = distmat, # geographic distance matrix  
  NH_radius = 700, # radius of neighbourhood (chosen a priori) to try  
  min_N = 5, # Minimum sample size to calculate metrics  
  metrics = c('GD'), # CWhat metrics to calculate  
)
```

```
sGD.800 <- sGD(genind, # genetic data  
  xy = latlong, # geographic coordinates  
  dist.mat = distmat, # geographic distance matrix  
  NH_radius = 800, # radius of neighbourhood (chosen a priori) to try  
  min_N = 5, # Minimum sample size to calculate metrics
```

```

    metrics = c('GD'), # CWhat metrics to calculate
  )

sGD.900 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 900, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
  #NHmat_ans = T,
  #genout_ans = T,
  #file_name = 'sGD.900'
)

sGD.1000 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1000, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1100 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1100, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1200 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1200, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1300 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1300, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

```

```
sGD.1500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)
```

```
sGD.1600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)
```

The sGD function is malfunctioning and does not give me Fis values, so I need to calculate them myself.

```
sGD.400$FIS <- 1 - sGD.400$Ho / sGD.400$Hs
sGD.500$FIS <- 1 - sGD.500$Ho / sGD.500$Hs
sGD.600$FIS <- 1 - sGD.600$Ho / sGD.600$Hs
sGD.700$FIS <- 1 - sGD.700$Ho / sGD.700$Hs
sGD.800$FIS <- 1 - sGD.800$Ho / sGD.800$Hs
sGD.900$FIS <- 1 - sGD.900$Ho / sGD.900$Hs
sGD.1000$FIS <- 1 - sGD.1000$Ho / sGD.1000$Hs
sGD.1100$FIS <- 1 - sGD.1100$Ho / sGD.1100$Hs
sGD.1200$FIS <- 1 - sGD.1200$Ho / sGD.1200$Hs
sGD.1300$FIS <- 1 - sGD.1300$Ho / sGD.1300$Hs
sGD.1400$FIS <- 1 - sGD.1400$Ho / sGD.1400$Hs
sGD.1500$FIS <- 1 - sGD.1500$Ho / sGD.1500$Hs
sGD.1600$FIS <- 1 - sGD.1600$Ho / sGD.1600$Hs
```

Label the neighbourhoods by the central individuals they are centred on.

```
sGD.400$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.500$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.600$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.700$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.800$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.900$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1000$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1100$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1200$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1300$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1400$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1500$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
sGD.1600$NH_ID <- latlong_spinosa_Dataset1_Nan_FSfilter$INDV
```

```
sGD_FIS <- c(sGD.400$FIS, sGD.500$FIS, sGD.600$FIS, sGD.700$FIS, sGD.800$FIS, sGD.900$FIS, sGD.1000$FIS,
sGD_radii <- rep(c(400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600), each = nrow
```

```

sgd_2sigma.spinosa_Dataset1_FSfilter <- tibble(sgd_rad, sgd_FIS)

ggplot(sgd_2sigma.spinosa_Dataset1_FSfilter,
       aes(y = sgd_FIS, x = as.factor(sgd_rad), fill = as.factor(sgd_rad))) +
  geom_boxplot(fill = 'grey') +
  geom_point() +
  geom_hline(yintercept = 0, colour = 'red') +
  guides(fill = FALSE) +
  theme_classic() +
  xlab('Neighbourhood Radius') +
  ylab(bquote( F[IS] )) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2))

ggplot(sgd_2sigma.spinosa_Dataset1_FSfilter,
       aes(x = as.factor(sgd_rad), y = sgd_FIS)) +
  geom_violin(fill = 'grey') +
  geom_point() +
  geom_hline(yintercept = 0, colour = 'red') +
  guides(fill = FALSE) +
  theme_classic() +
  xlab('Neighbourhood Radius') +
  ylab(bquote( F[IS] )) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2))

summary(sgd.400$FIS)
summary(sgd.500$FIS)
summary(sgd.600$FIS)
summary(sgd.700$FIS)
summary(sgd.800$FIS)
summary(sgd.900$FIS)
summary(sgd.1000$FIS)
summary(sgd.1100$FIS)
summary(sgd.1200$FIS)
summary(sgd.1300$FIS)
summary(sgd.1400$FIS)
summary(sgd.1500$FIS)
summary(sgd.1600$FIS)

```

genind_spinosa_Dataset1_Oak_wild_FSfilter

```

# Set which dataset, latlongs, and distance matrix I'll be using.

genind <- genind_spinosa_Dataset1_Oak_wild_FSfilter

```

```
latlong <- latlong_spinosa_Dataset1_Oak_FSfilter
distmat <- distmat_spinosa_Dataset1_Oak_FSfilter
```

```
sGD.400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.700 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 700, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.800 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 800, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)
```

```
sGD.900 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 900, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
  #NHmat_ans = T,
```

```

    #genout_ans = T,
    #file_name = 'sGD.900'
  )

sGD.1000 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1000, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1100 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1100, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1200 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1200, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1300 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1300, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1400 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1400, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

sGD.1500 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1500, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # CWhat metrics to calculate
)

```

```

sGD.1600 <- sGD(genind, # genetic data
  xy = latlong, # geographic coordinates
  dist.mat = distmat, # geographic distance matrix
  NH_radius = 1600, # radius of neighbourhood (chosen a priori) to try
  min_N = 5, # Minimum sample size to calculate metrics
  metrics = c('GD'), # What metrics to calculate
)

# The sGD function is malfunctioning and does not give me Fis values, so I need to calculate them myself.

sGD.400$FIS <- 1 - sGD.400$Ho / sGD.400$Hs
sGD.500$FIS <- 1 - sGD.500$Ho / sGD.500$Hs
sGD.600$FIS <- 1 - sGD.600$Ho / sGD.600$Hs
sGD.700$FIS <- 1 - sGD.700$Ho / sGD.700$Hs
sGD.800$FIS <- 1 - sGD.800$Ho / sGD.800$Hs
sGD.900$FIS <- 1 - sGD.900$Ho / sGD.900$Hs
sGD.1000$FIS <- 1 - sGD.1000$Ho / sGD.1000$Hs
sGD.1100$FIS <- 1 - sGD.1100$Ho / sGD.1100$Hs
sGD.1200$FIS <- 1 - sGD.1200$Ho / sGD.1200$Hs
sGD.1300$FIS <- 1 - sGD.1300$Ho / sGD.1300$Hs
sGD.1400$FIS <- 1 - sGD.1400$Ho / sGD.1400$Hs
sGD.1500$FIS <- 1 - sGD.1500$Ho / sGD.1500$Hs
sGD.1600$FIS <- 1 - sGD.1600$Ho / sGD.1600$Hs

# Label the neighbourhoods by the central individuals they are centred on.
sGD.400$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.500$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.600$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.700$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.800$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.900$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1000$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1100$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1200$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1300$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1400$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1500$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV
sGD.1600$NH_ID <- latlong_spinosa_Dataset1_Oak_FSfilter$INDV

sGD_FIS <- c(sGD.400$FIS, sGD.500$FIS, sGD.600$FIS, sGD.700$FIS, sGD.800$FIS, sGD.900$FIS, sGD.1000$FIS,
sGD_radII <- rep(c(400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600), each = nrow
sGD_2sigma.spinosa_Dataset1_Oak_FSfilter <- tibble(sGD_radII, sGD_FIS)

ggplot(sGD_2sigma.spinosa_Dataset1_Oak_FSfilter,
  aes(y = sGD_FIS, x = as.factor(sGD_radII), fill = as.factor(sGD_radII))) +
  geom_boxplot(fill = 'grey') +
  geom_point() +
  geom_hline(yintercept = 0, colour = 'red') +
  guides(fill = FALSE) +
  theme_classic() +

```

```

xlab('Neighbourhood Radius') +
ylab(bquote( F[IS])) +
theme(
  # Make axis titles bold
  axis.title = element_text(face = 2))

ggplot(sGD_2sigma.spinosa_Dataset1_Oak_FSfilter,
  aes(x = as.factor(sGD_radrii), y = sGD_FIS)) +
  geom_violin(fill = 'grey')+
  geom_point()+
  geom_hline(yintercept = 0, colour = 'red')+
  guides(fill = FALSE) +
  theme_classic()+
  xlab('Neighbourhood Radius') +
  ylab(bquote( F[IS])) +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2))

summary(sGD.400$FIS)
summary(sGD.500$FIS)
summary(sGD.600$FIS)
summary(sGD.700$FIS)
summary(sGD.800$FIS)
summary(sGD.900$FIS)
summary(sGD.1000$FIS)
summary(sGD.1100$FIS)
summary(sGD.1200$FIS)
summary(sGD.1300$FIS)
summary(sGD.1400$FIS)
summary(sGD.1500$FIS)
summary(sGD.1600$FIS)

```

8. Effective Population Size

Effective Population Size:

This analysis aims to produce estimates of Effective Population Size (N_e) for both populations. The effect of the presence of siblings will also be assessed (and the effect of sample size is assessed in the next section). To reduce the effect of linkage disequilibrium from physical linkage, the dataset is thinned to several random sets of 1000 SNPs before being run, which is a number of SNPs considered to be sufficient for good estimates (Do et al., 2014). I only select polymorphic SNPs because monomorphic SNPs are uninformative in this analysis (correlation of 1).

The method of N_e estimation used will be the linkage disequilibrium method as implemented in Do et al., (2014), which is a widely used method whose assumptions have been thoroughly assessed (although there is still a lot to understand about those assumptions). I will be using the implementation in NeEstimator2 through a dartR convenience function.

To save time and recreate the main plot using the data I used, skip to the sub-heading ‘Short-cut: use saved data’.

File path for Ne estimator

Specify the file path used for Ne estimator v2. The `gl.LDNe()` function requires an installation of this software, and the directory containing the executable must be directly referenced in the code.

Note that you may have to rename the `Ne2.exe` into `Ne2-1.exe` for it to be recognised by the `dartR` function.

```
# The path in OneDrive
neest.path1 <- '~/Software/neestimator'
```

Estimate Ne for full populations

spinosa_Dataset1_Nan

Subsample 1000 SNPs from the dataset to try to reduce physical linkage and generate 5 new random subsets to analyse.

```
genlight_spinosa_Dataset1_Nan_rep1 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan, n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_rep2 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan, n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_rep3 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan, n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_rep4 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan, n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_rep5 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan, n = 1000, method = "random", mono.rm = TRUE)
```

Calculate Ne for each of the replicate datasets.

```
LDNe.spinosa_Dataset1_Nan_rep1 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_rep1,
  outfile = 'LDNe_spinosa_Dataset1_Nan_rep1',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE, # There are no singletons here but I leave it as default
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_rep2 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_rep2,
  outfile = 'LDNe_spinosa_Dataset1_Nan_rep2'
  outpath = 'output',
```

```

neest.path = neest.path1,
critical = c(0, 0.01, 0.02, 0.05),
singleton.rm = TRUE,
mating = 'random',
plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_rep3 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_rep3,
  outfile = 'LDNe_spinosa_Dataset1_Nan_rep3',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_rep4 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_rep4,
  outfile = 'LDNe_spinosa_Dataset1_Nan_rep4',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_rep5 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_rep5,
  outfile = 'LDNe_spinosa_Dataset1_Nan_rep5',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

```

spinosa_Dataset1_Nan_FSfilter

Subsample 1000 SNPs from the dataset to try to reduce physical linkage and generate 5 new random subsets to analyse.

```

genlight_spinosa_Dataset1_Nan_FSfilter_rep1 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_FSfilter_rep2 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

```

```

genlight_spinosa_Dataset1_Nan_FSfilter_rep3 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_FSfilter_rep4 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Nan_FSfilter_rep5 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Nan_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

```

Calculate Ne for each of the replicate datasets.

```

LDNe.spinosa_Dataset1_Nan_FSfilter_rep1 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_FSfilter_rep1,
  outfile = 'LDNe_spinosa_Dataset1_Nan_FSfilter_rep1',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_FSfilter_rep2 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_FSfilter_rep2,
  outfile = 'LDNe_spinosa_Dataset1_Nan_FSfilter_rep2',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_FSfilter_rep3 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_FSfilter_rep3,
  outfile = 'LDNe_spinosa_Dataset1_Nan_FSfilter_rep3',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_FSfilter_rep4 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_FSfilter_rep4,
  outfile = 'LDNe_spinosa_Dataset1_Nan_FSfilter_rep4',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),

```

```

singleton.rm = TRUE,
mating = 'random',
plot.out = TRUE
)

LDNe.spinosa_Dataset1_Nan_FSfilter_rep5 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Nan_FSfilter_rep5,
  outfile = 'LDNe_spinosa_Dataset1_Nan_FSfilter_rep5',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

```

spinosa_Dataset1_Oak

Subsample 1000 SNPs from the dataset to try to reduce physical linkage and generate 5 new random subsets to analyse.

```

genlight_spinosa_Dataset1_Oak_wild_rep1 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild, n = 1000,
  method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_rep2 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild, n = 1000,
  method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_rep3 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild, n = 1000,
  method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_rep4 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild, n = 1000,
  method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_rep5 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild, n = 1000,
  method = "random", mono.rm = TRUE)

```

Calculate Ne for each of the replicate datasets.

```

LDNe.spinosa_Dataset1_Oak_wild_rep1 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_rep1,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_rep1',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

```

```

)

LDNe.spinosa_Dataset1_Oak_wild_rep2 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_rep2,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_rep2',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Oak_wild_rep3 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_rep3,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_rep3',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Oak_wild_rep4 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_rep4,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_rep4',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Oak_wild_rep5 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_rep5,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_rep5',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

```

spinosa_Dataset1_Oak_wild_FSfilter

Subsample 1000 SNPs from the dataset to try to reduce physical linkage and generate 5 new random subsets to analyse.

```

genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep1 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep2 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep3 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep4 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep5 <- gl.subsample.loci(
  genlight_spinosa_Dataset1_Oak_wild_FSfilter,
  n = 1000, method = "random", mono.rm = TRUE)

```

Calculate Ne for each of the replicate datasets.

```

LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep1 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep1,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_FSfilter_rep1',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep2 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep2,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_FSfilter_rep2',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep3 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep3,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_FSfilter_rep3',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

```

```

)

LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep4 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep4,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_FSfilter_rep4',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep5 <- gl.LDNe(
  x = genlight_spinosa_Dataset1_Oak_wild_FSfilter_rep5,
  outfile = 'LDNe_spinosa_Dataset1_Oak_wild_FSfilter_rep5',
  outpath = 'output',
  neest.path = neest.path1,
  critical = c(0, 0.01, 0.02, 0.05),
  singleton.rm = TRUE,
  mating = 'random',
  plot.out = TRUE
)

```

Plotting

Data entry

Before I can plot anything I have to extract all of the data from the Ne runs above and enter them into a dataframe with the right format. This is an overly complicated approach and will be the hardest component to transfer to other datasets. Consider other alternatives if using this to guide your analysis.

```

# Create the parameter values that will fill the table
pop <- c('Nangur', 'Oakview')
sib_status <- c('Full-sibs remain', 'Full-sibs removed')
allele_freq <- c(0.05, 0.01)
rep <- c(1:5, 'mean')
Ne_estimate <- NA
CI_low <- NA
CI_high <- NA

# Generate a table from the above variables
Ne.tab <- expand.grid(pop = pop, sib_status = sib_status,
                     allele_freq = allele_freq, rep = rep,
                     Ne_estimate = Ne_estimate,
                     CI_low = CI_low, CI_high = CI_high)

# Compile all relevant results into a list
# following the order of the above table.

Ne.list <- list(LDNe.spinosa_Dataset1_Nan_rep1,

```

```

LDNe.spinosa_Dataset1_Oak_wild_rep1,
LDNe.spinosa_Dataset1_Nan_FSfilter_rep1,
LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep1,

LDNe.spinosa_Dataset1_Nan_rep2,
LDNe.spinosa_Dataset1_Oak_wild_rep2,
LDNe.spinosa_Dataset1_Nan_FSfilter_rep2,
LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep2,

LDNe.spinosa_Dataset1_Nan_rep3,
LDNe.spinosa_Dataset1_Oak_wild_rep3,
LDNe.spinosa_Dataset1_Nan_FSfilter_rep3,
LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep3,

LDNe.spinosa_Dataset1_Nan_rep4,
LDNe.spinosa_Dataset1_Oak_wild_rep4,
LDNe.spinosa_Dataset1_Nan_FSfilter_rep4,
LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep4,

LDNe.spinosa_Dataset1_Nan_rep5,
LDNe.spinosa_Dataset1_Oak_wild_rep5,
LDNe.spinosa_Dataset1_Nan_FSfilter_rep5,
LDNe.spinosa_Dataset1_Oak_wild_FSfilter_rep5)

# Create for loop to enter data into the table from the list of results.
# There are two from each list (for the two minor allele frequencies)

# Iterate through the 5 groups of 4 (nee 6) rows of Ne.tab
for (i in c(1, 9, 17, 25, 33)){
  for (j in 0:3){
    # Iterate through each of the 4 rows in each of 5 groups of rows.
    if (i == 9){
      # For the value of the row of Ne.tab, choose the appropriate list
      X = 4
    } else if (i == 17){
      X = 8
    } else if (i == 25){
      X = 12
    } else if (i == 33){
      X = 16
    } else {
      X = 0
    }
  }

  print (paste0('Row of Ne.tab for frequency (0.05) is:', i+j,
    ' | Row of Ne.tab for frequency (0.01) is:', i+j+4,
    ' | Ne.list item is:', i+j-X))
}

```

```

# Allele frequency = 0.05

# Enter the Ne estimate for the ith list item, use the frequency 0.05.
Ne.tab$Ne_estimate[i+j] <- Ne.list[[i+j-X]]$pop1$`Frequency 1`[6]

# Enter the lower jackknife confidence interval for the ith list item,
# use the frequency 0.05.
Ne.tab$CI_low[i+j] <- Ne.list[[i+j-X]]$pop1$`Frequency 1`[9]

# Enter the upper CI jackknife estimate for the ith list item,
# use the frequency 0.05.
Ne.tab$CI_high[i+j] <- Ne.list[[i+j-X]]$pop1$`Frequency 1`[10]

# Allele frequency = 0.01.

# Enter the i+6th Ne estimate for the ith list item, use the frequency 0.01.
Ne.tab$Ne_estimate[i+j+4] <- Ne.list[[i+j-X]]$pop1$`Frequency 3`[6]

# Enter the i+6th lower jackknife confidence interval for the ith list item,
# use the frequency 0.01.
Ne.tab$CI_low[i+j+4] <- Ne.list[[i+j-X]]$pop1$`Frequency 3`[9]

# Enter the i+6th upper CI jackknife estimate for the ith list item,
# use the frequency 0.01.
Ne.tab$CI_high[i+j+4] <- Ne.list[[i+j-X]]$pop1$`Frequency 3`[10]

}

}

# Now collect the mean Ne values where...
# Row 61 takes the mean of rows 1, 13, 25, 37, 49
# Row 62 takes the mean of rows 2, 14, 26, 38, 50
# ...
# Row 72 takes the mean of rows 12, 24, 36, 48, 60.
Ne.tab$SE <- NA

# A for loop that iterates through the 8 means I have to calculate.
for (i in 0:7){

# Calculate the mean for each parameter combination.
Ne.tab$Ne_estimate[i + 41] <- mean(as.numeric(c(Ne.tab$Ne_estimate[i+1],
                                                Ne.tab$Ne_estimate[i+9],
                                                Ne.tab$Ne_estimate[i+17],
                                                Ne.tab$Ne_estimate[i+25],
                                                Ne.tab$Ne_estimate[i+33])))

# Calculate the SE for each parameter combination.
Ne.tab$SE[i + 41] <- sd(as.numeric(c(Ne.tab$Ne_estimate[i+1],
                                      Ne.tab$Ne_estimate[i+9],
                                      Ne.tab$Ne_estimate[i+17],
                                      Ne.tab$Ne_estimate[i+25],

```

```

Ne.tab$Ne_estimate[i+33])))/sqrt(5)
}

# Set variables as numerical vectors.
Ne.tab$Ne_estimate <- as.numeric(Ne.tab$Ne_estimate)

Ne.tab$CI_low <- as.numeric(Ne.tab$CI_low)

Ne.tab$CI_high <- as.numeric(Ne.tab$CI_high)

#Ne.tab$rep <- as.numeric(Ne.tab$rep)

```

Short-cut: use saved data

To save time and recreate the main plot using the data I used, run the following chunk before plotting.

```

Ne.tab <- read.csv(
  file = '../data/other_files/Ne_tab.csv')

# Update the wording of the sib filter
Ne.tab$sib_status -> Ne.tab$relationship_filter

Ne.tab$relationship_filter[Ne.tab$relationship_filter == 'Full-sibs remain'] <- 'First-degree relations
Ne.tab$relationship_filter[Ne.tab$relationship_filter == 'Full-sibs removed'] <- 'First-degree relations

# Update the MAF label
Ne.tab$allele_freq[Ne.tab$allele_freq == 0.01] <- 'MAF > 0.01'
Ne.tab$allele_freq[Ne.tab$allele_freq == 0.05] <- 'MAF > 0.05'

```

ggplot

Nangur

```

# Extract the means so they are a separate value.
Ne_means_Nangur <- Ne.tab[Ne.tab$rep == 'mean' & Ne.tab$pop == 'Nangur' ,]

# Change the # rep fo the means to be an x axis value (where I want to put it)
Ne_means_Nangur$rep <- rep(2.5, times = length(Ne_means_Nangur$rep))

# Plot
Ne.tab[Ne.tab$rep != 'mean' & Ne.tab$pop == 'Nangur' ,] %>%
  mutate(rep = as.numeric(rep)) %>%
  ggplot(aes(x = rep, y = Ne_estimate)) +
    geom_point() +
    geom_errorbar(aes(ymin = CI_low,
                      ymax = CI_high),
                  width=0) +

```

```

facet_grid( allele_freq ~ sib_status) +
theme_bw() +
theme(axis.text.x = element_blank(),
      axis.title.x = element_blank(),
      axis.ticks.x = element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank())+
xlim(-3, 11) +
ylab(bquote(N[' E'])) +
geom_point(data = Ne_means_Nangur,
          aes(x = rep, y = Ne_estimate),
          alpha = 0.5,
          size = 3,
          colour = 'forestgreen') +
geom_errorbar(data = Ne_means_Nangur,
             aes(ymin = Ne_estimate - SE,
                 ymax = Ne_estimate + SE),
             colour = 'forestgreen') +
geom_text(data = Ne_means_Nangur,
          mapping = aes(x = -Inf,
                        y = -Inf,
                        label = paste0('mean = ', as.character(Ne_estimate))),
          x = 7, y = 125,
          colour = 'forestgreen',
          size = 3,
          fontface = 'bold') +
theme(strip.background = element_rect(colour = "black",
                                       fill = alpha("forestgreen",0.2) ))

```

Oakview

```

# Extract the means so they are a separate value.
Ne_means_Oakview <- Ne.tab[Ne.tab$rep == 'mean' & Ne.tab$pop == 'Oakview' ,]

# Change the # rep fo the means to be an x axis value (where I want to put it)
Ne_means_Oakview$rep <- rep(2.5, times = length(Ne_means_Oakview$rep))

# Plot
Ne.tab[Ne.tab$rep != 'mean' & Ne.tab$pop == 'Oakview' ,] %>%
  mutate(rep = as.numeric(rep)) %>%
ggplot(aes(x = rep, y = Ne_estimate)) +
  geom_point()+
  geom_errorbar(aes(ymin = CI_low,
                   ymax = CI_high),
               width=0) +
facet_grid( allele_freq ~ sib_status) +
theme_bw() +
theme(axis.text.x = element_blank(),
      axis.title.x = element_blank(),
      axis.ticks.x = element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank())+

```

```

xlim(-3, 11) +
ylab(bquote(N[' E'])) +
scale_y_continuous(n.breaks = 8 ) +
geom_point(data = Ne_means_Oakview,
  aes(x = rep, y = Ne_estimate),
  alpha = 0.5,
  size = 3,
  colour = 'royalblue') +
geom_errorbar(data = Ne_means_Oakview,
  aes(ymin = Ne_estimate - SE,
    ymax = Ne_estimate + SE),
  colour = 'royalblue') +
geom_text(data = Ne_means_Oakview,
  mapping = aes(x = -Inf,
    y = -Inf,
    label = paste0('mean = ', as.character(Ne_estimate))),
  x = 7, y = 600,
  colour = 'royalblue',
  size = 3,
  fontface = 'bold') +
theme(strip.background = element_rect(colour = "black",
  fill = alpha("royalblue",0.2) ))

```

Combined Plot

Prepare dataframe

```

# Subset the dataframe to have just the replicates of the analysis (not means)
Ne.tab[Ne.tab$rep != 'mean',] %>%

# Make the replicates into a numeric vector, and assign to new dataframe
mutate(rep = as.numeric(rep)) -> Ne.tab.reps

# Re-number the Oakview reps so that they are
# graphically displaced relative to Netview
Ne.tab.reps$rep[Ne.tab.reps$pop == 'Oakview'] <- Ne.tab.reps$rep[
  Ne.tab.reps$pop == 'Oakview'] + 6

```

Now plot...

```

# Plot
Ne.tab.reps |>
ggplot(aes(x = rep, y = Ne_estimate)) +
  geom_point()+
  geom_errorbar(aes(ymin = CI_low,
    ymax = CI_high),
    width = 0) +
  facet_grid( allele_freq ~ relationship_filter) +
  theme_bw() +
  theme(axis.text.x = element_blank(),
    axis.title.x = element_blank(),
    axis.ticks.x = element_blank(),
    panel.grid.major = element_blank(),

```

```

    panel.grid.minor = element_blank(),
    axis.title = element_text(face = 2))+
xlim(0, 12) +
ylab(bquote(N[ ' e' ])) +
#coord_cartesian(ylim = c(0, 250))+
scale_y_continuous(n.breaks = 6) +
# Oakview means
geom_point(data = Ne_means_Oakview,
  aes(x = rep+7, y = Ne_estimate),
  alpha = 0.5,
  size = 3,
  colour = 'royalblue') +

# Oakview mean standard error bars
geom_errorbar(data = Ne_means_Oakview,
  aes(x = rep + 7,
    ymin = Ne_estimate - SE,
    ymax = Ne_estimate + SE),
  colour = 'royalblue') +

# Oakview mean text
geom_text(data = Ne_means_Oakview,
  mapping = aes(x = -Inf,
    y = -Inf,
    label = paste0('mean = ', as.character(Ne_estimate))),
  x = 9.5, y = 5,
  colour = 'royalblue',
  size = 3,
  fontface = 'bold') +

# Nangur means
geom_point(data = Ne_means_Nangur,
  aes(x = rep, y = Ne_estimate),
  alpha = 0.5,
  size = 3,
  colour = 'forestgreen') +

# Nangur mean standard error Bars
geom_errorbar(data = Ne_means_Nangur,
  aes(ymin = Ne_estimate - SE,
    ymax = Ne_estimate + SE),
  colour = 'forestgreen') +

# Nangur mean text
geom_text(data = Ne_means_Nangur,
  mapping = aes(x = -Inf,
    y = -Inf,
    label = paste0('mean = ', as.character(Ne_estimate))),
  x = 2.5, y = 5,
  # alternative positions that sit above se bars
  # c(70, 100, 140, 80, 110, 170)
  colour = 'forestgreen',
  size = 3,

```

```

fontface = 'bold') +

# Midline
geom_vline(xintercept = 6.0, colour = 'grey', linetype = 'dotted', size = 1)

```

8b. Effective Population Size (effect of sample size)

Here I investigate the effect of sample size on Ne estimates. As with the diversity and inbreeding section, there is a limitation to my approach. These subsamples come from the same genlights, an assumption is that a consistent set of loci is being analysed across all subsamples, rather than a new set of loci generated for each subsample.

In each subsection, I use a for loop to subsample a genlight 100 times at different sizes and I calculate Ne for each subsample. I then plot the results.

To save time and recreate the main plot using the data I used, skip to the sub-heading ‘Short-cut: use saved data’.

File path for Ne estimator

Specify the file path used for Ne estimator v2. The gl.LDNE() function requires an installation of this software, and the directory containing the executable must be directly referenced in the code.

Note that you may have to rename the Ne2.exe into Ne2-1.exe for it to be recognised by the dartR function.

```

# The path in OneDrive
neest.path1 <- '~/Software/neestimator'

```

spinosa_Dataset1_Oak

```

nrep = c(1:100)

sample.size <- c(seq(from = 15, to = 45, by = 5))

Ne_estimate <- NA

CI_low <- NA

CI_high <- NA

Ne.samplesize.results.Oakview <- expand.grid(nrep = nrep,
                                              sample.size = sample.size,
                                              Ne_estimate = Ne_estimate,
                                              CI_low = CI_low,
                                              CI_high = CI_high)

for (i in 1:length(Ne.samplesize.results.Oakview$nrep)){

  # Take a subsample of size i o individuals

```

```

RESULT.i <- gl.sample(genlight_spinosa_Dataset1_Oak_wild,
                      nsample = Ne.samplesize.results.Oakview$sample.size[i],
                      replace = FALSE, verbose = 0) |>

  # Required processing of dataset
  gl.compliance.check(verbose = 0) |>

  # Take a subsample of 1000 SNPs to reduce LD
  gl.subsample.loci(n = 1000, method = "random", mono.rm = TRUE) |>

  # Estimate Ne
  gl.LDNe(

    outfile = 'LDNe_spinosa_Dataset1_Oak_wild_subsample',
                                outpath = 'output',
                                neest.path = neest.path1,
                                critical = c(0.05),
                                singleton.rm = TRUE,
                                mating = 'random',
                                plot.out = TRUE
  )

  # Enter the Ne estimate for the run
  Ne.samplesize.results.Oakview$Ne_estimate[i] <- RESULT.i$pop1$`Frequency 1`[6]

  # Enter the lower jackknife confidence interval for the run,
  # use the frequency 0.05.
  Ne.samplesize.results.Oakview$CI_low[i] <- RESULT.i$pop1$`Frequency 1`[9]

  # Enter the upper CI jackknife estimate for the run, use the frequency 0.05.
  Ne.samplesize.results.Oakview$CI_high[i] <- RESULT.i$pop1$`Frequency 1`[10]

}

```

```

Ne.samplesize.results.Oakview$Ne_estimate <-
  as.numeric(Ne.samplesize.results.Oakview$Ne_estimate)

Ne.samplesize.results.Oakview[
  Ne.samplesize.results.Oakview$Ne_estimate != Inf,] |>
  ggplot(aes(y = as.factor(sample.size), x = Ne_estimate)) +
  geom_density_ridges(fill = 'royalblue',
                     alpha = 0.6,
                     quantile_lines = TRUE,
                     quantiles = c(0.05, 0.5, 0.95)) +
  geom_point() +
  theme_classic() +
  xlab(bquote(N[' E'])) +
  scale_y_discrete('Size of subsample') +
  theme(
    # Make axis titles bold

```

```
axis.title = element_text(face = 2)) +
scale_x_continuous(n.breaks = 8)
```

spinoso_Dataset1_Nan

```
nrep = c(1:100)

sample.size <- c(seq(from = 15, to = 25, by = 5))

Ne_estimate <- NA

CI_low <- NA

CI_high <- NA

Ne.samplesize.results.Nangur <- expand.grid(nrep = nrep,
                                             sample.size = sample.size,
                                             Ne_estimate = Ne_estimate,
                                             CI_low = CI_low,
                                             CI_high = CI_high)

for (i in 1:length(Ne.samplesize.results.Nangur$nrep)){

  # Take a subsample of size i o individuals
  RESULT.i <- gl.sample(genlight_spinoso_Dataset1_Nan,
                        nsample = Ne.samplesize.results.Nangur$sample.size[i],
                        replace = FALSE, verbose = 0) |>

  # Required processing of dataset
  gl.compliance.check(verbose = 0) |>

  # Take a subsample of 1000 SNPs to reduce LD
  gl.subsample.loci(n = 1000, method = "random", mono.rm = TRUE) |>

  # Estimate Ne
  gl.LDNe(
    outfile = 'LDNe_spinoso_Dataset1_Nan_subsample',
    outpath = 'output',
    neest.path = neest.path1,
    critical = c(0.05),
    singleton.rm = TRUE,
    mating = 'random',
    plot.out = TRUE
  )

  # Enter the Ne estimate for the run
  Ne.samplesize.results.Nangur$Ne_estimate[i] <- RESULT.i$pop1$`Frequency 1`[6]

  # Enter the lower jackknife confidence interval for the run,
  # use the frequency 0.05.
  Ne.samplesize.results.Nangur$CI_low[i] <- RESULT.i$pop1$`Frequency 1`[9]
```

```

# Enter the upper CI jackknife estimate for the run, use the frequency 0.05.
Ne.samplesize.results.Nangur$CI_high[i] <- RESULT.i$pop1$`Frequency 1`[10]

}

```

```

Ne.samplesize.results.Nangur$Ne_estimate <- as.numeric(
  Ne.samplesize.results.Nangur$Ne_estimate)

Ne.samplesize.results.Nangur[
  Ne.samplesize.results.Nangur$Ne_estimate != Inf,] |>
  ggplot(aes(y = as.factor(sample.size), x = Ne_estimate)) +
  geom_density_ridges(fill = 'forestgreen',
    alpha = 0.6,
    quantile_lines = TRUE,
    quantiles = c(0.05, 0.5, 0.95)) +
  geom_point() +
  theme_classic() +
  xlab(bquote(N[' E'])) +
  scale_y_discrete('Size of subsample') +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
  scale_x_continuous(n.breaks = 8)

```

Short-cut: use saved data

To save time and recreate the main plot using the data I used, run the following chunk before plotting.

```

#Ne.samplesize.results.Nangur <- read.csv(
#file =
# '../data/other_files/Ne_samplesize_results_Nangur.csv')

#Ne.samplesize.results.Oakview <- read.csv(
#file =
# '../data/other_files/Ne_samplesize_results_Oakview.csv')

```

Combined plot

Merge the Oakview and Nangur dataframes into a single plot.

```

# Add a pop column
Ne.samplesize.results.Nangur$POP <-
  'Nangur'

Ne.samplesize.results.Oakview$POP <-
  'Oakview'

# Join the two dataframes

```

```
Ne.samplesize.results.joined <- rbind(
  Ne.samplesize.results.Nangur,
  Ne.samplesize.results.Oakview)
```

Plot

```
Ne.samplesize.results.joined |>
  ggplot(aes(y = as.factor(sample.size),
    x = Ne_estimate, fill = POP)) +
  geom_density_ridges(alpha = 0.6,
    quantile_lines = TRUE,
    quantiles = c(0.05, 0.5, 0.95),
    scale = 0.9) +
  scale_fill_manual(values = c(
    'forestgreen', 'royalblue')) +
  geom_point(aes(colour = POP), shape = '|', size = 2) +
  scale_colour_manual(values = c(
    'lightgreen', 'darkblue')) +
  theme_classic() +
  xlab(bquote(N['e'])) +
  scale_y_discrete('Size of subsample') +
  theme(
    # Make axis titles bold
    axis.title = element_text(face = 2)) +
  scale_x_continuous(n.breaks = 8) +
  labs(fill = 'Population') +
  guides(colour = FALSE)
```

9. Mapping

Mapping:

The purpose of this Rmarkdown is to present various analyses on a map, using latitude and longitude, and showing a scale bar. An important component of this map is that the latitudes and longitudes will be obscured and the background map will be blank. This is so that poachers and herpers are unable to find the location of this species within the national parks. This species is very fragile and needs to be protected from both illegal, destructive activities and unnecessary disturbance.

Nangur and Oakview maps will be on the same map and to scale, but with axes cut so as to show the distance between them. This uses the package ggbreak.

There will be three maps. - One will be just geography, with colours used to cross reference with the netviewR plots. - One will show MLH values. - One will show related individuals.

An additional reference map showing the locations and shapes of the national parks in reference to Australia/SE Queensland.

‘NetviewR’ map

```
Figure2b <- NanguraSpinosa.dat[NanguraSpinosa.dat$Year != '2006-2007',] |>
  ggplot(
```

```

# Choose aesthetic mapping parameters
aes(x = Longitude, y = Latitude, fill = Sample_cluster,
     colour = captive_wild)) +

# Choose to use a scatter plot, with shape parameters
geom_point(size = 3, shape = 21, stroke = 1.5) +
theme_classic() +

# Choose fill and colours for each level of the variable
scale_fill_manual(values = c('yellow', 'olivedrab', 'olivedrab1',
                             'forestgreen', 'green', 'navy',
                             'darkslategray1', 'royalblue', 'deepskyblue',
                             'mediumpurple1', 'mediumorchid1')) +
scale_colour_manual(values = c('darkorange', 'black')) +

# Break the x axis
scale_x_break(breaks = c(0.0176511, 0.3166511), scales = 3) +

# Choose x axis ticks to show, this somehow also duplicates it
scale_x_continuous(
  breaks = c(-0.0033489, 0.0166511, 0.3266511, 0.3466511, 0.3666511)) +

# Duplicate y axis on the right
scale_y_continuous(sec.axis = dup_axis()) +

# Edit the legend
labs(colour = 'Status') +
guides(fill = FALSE) +

theme(
  # Remove axis tick labels for confidentiality
  axis.text = element_blank(),

  # Make axis titles bold
  axis.title = element_text(face = 2)) +

# Label geographic locations
annotate("text", x = 0.008, y = 0.00166043,
         label = 'Nangur', fontface = 2, size = 5) +
annotate("text", x = 0.3366511, y = -0.01833957,
         label = 'Oakview', fontface = 2, size = 5)

```

Figure2b

```

citation('ggbreak')

```

```

# Scale:
# X axis ticks are 0.02 degrees longitude.
# Y axis ticks are 0.01 degrees latitude.
# Thus, X axis marks are approximately (very close to) 2 km apart.
# Y axis marks are approximately 1.11 km apart.

```

MLH (inbreeding) map

Here I calculate individual multiple-locus heterozygosity values. This is just the proportion of SNPs that are heterozygous..

```
MLH_spinosa_Dataset2 <- genlight_spinosa_Dataset2 |>
  gl.report.heterozygosity(
    method = "ind",
    n.invariant = 3730497,
    plot.out = TRUE,
  )
```

Merge the results with metadata so that I can view what population they come from and associate other metadata from that individual with its MLH.

```
# Rename ind.name INDV

colnames(MLH_spinosa_Dataset2)[1] <- 'INDV'

# Merge the MLH values with NanguraSpinosa.dat

MLH_spinosa_Dataset2 <- merge(x = MLH_spinosa_Dataset2, y = NanguraSpinosa.dat,
                             by = 'INDV')
```

Plot

```
Figure5b <- MLH_spinosa_Dataset2[MLH_spinosa_Dataset2$Year != '2006-2007',] |>
  ggplot(

    # Choose aesthetic mapping parameters
    aes(x = Longitude, y = Latitude, colour = Ho)) +

    # Choose to use a scatter plot, with shape parameters
    #geom_point(size = 3) +

    # Choose to use a scatter plot, with size parameters.
    # Include a slight jitter so that all individual points can be seen.
    # Set a random seed so that the jitter is reproducible
    geom_point(position = position_jitter(width = 0.001,
                                           height = 0.001, seed = 41), size = 2) +

    theme_classic() +

    # Break the x axis
    scale_x_break(breaks = c(0.0176511, 0.3166511), scales = 3) +

    # Choose x axis ticks to show, this somehow also duplicates it
    scale_x_continuous(
      breaks = c(-0.0033489, 0.0166511, 0.3266511, 0.3466511, 0.3666511)) +
```

```

# Duplicate y axis on the right
scale_y_continuous(sec.axis = dup_axis()) +

# Set fill scale for MLH, and reverse it
scale_colour_viridis(option = 'plasma',
                     direction = -1) +

# Edit the legend
labs(colour = 'Individual MLH') +
#guides(colour = FALSE) + # This removes the legend, which is necessary for
# plotting the figure for the paper.

theme(
  # Remove axis tick labels for confidentiality
  axis.text = element_blank(),

  # Make axis titles bold
  axis.title = element_text(face = 2)) +

# Label geographic locations
annotate("text", x = 0.008 , y = 0.00166043,
         label = 'Nangur', fontface = 2, size = 5) +
annotate("text", x = 0.3366511, y = -0.01833957,
         label = 'Oakview', fontface = 2, size = 5)

```

Figure5b

```

# Scale:
# X axis ticks are 0.02 degrees longitude.
# Y axis ticks are 0.01 degrees latitude.
# Thus, X axis marks are approximately (very close to) 2 km apart.
# Y axis marks are approximately 1.11 km apart.

```

Related individuals map

Here I plot individuals that are related to each other in fullsib families as described in the paper. The groupings are identified by colour. E.g., The two parents individual would be grouped together with their child (if they bred in the wild). Light grey individuals are not part of these groups.

Redefine families I first need to redefine individuals that should be grouped together on the map. Specifically, two unrelated individuals that bred in captivity should not be marked as relatives on the map, but if they bred in the wild that is okay.

```

# Remove full-sib families 15, 6, 5, and 4, as they come from pairings
# that were created in captivity.

NanguraSpinosa.dat$Family_FS[ NanguraSpinosa.dat$Family_FS == 15 |
                             NanguraSpinosa.dat$Family_FS == 6 |

```

```

NanguraSpinosa.dat$Family_FS == 5 |
NanguraSpinosa.dat$Family_FS == 4 |
NanguraSpinosa.dat$Family_FS == ""] <- NA |>
as.numeric()

# Remove half-sib families 15, 5, and 4. Family 6 is partly correct,
# but requires the removal of individual BSU0468 who is only related through
# offspring that were produced in captivity.

NanguraSpinosa.dat$Family_HS[ NanguraSpinosa.dat$Family_HS == 15 |
  NanguraSpinosa.dat$Family_HS == 5 |
  NanguraSpinosa.dat$Family_HS == 4 |
  NanguraSpinosa.dat$Family_HS == ""] <- NA |>
as.numeric()

NanguraSpinosa.dat$Family_HS[NanguraSpinosa.dat$INDV == 'BSU0468'] <- NA

```

Fullsibs To be specific, individuals are the same colour as other individuals that they are fullsibs with (identified in Colony), or individuals that are fullsibs with their fullsibs.

```

map3 <- NanguraSpinosa.dat[NanguraSpinosa.dat$Year != '2006-2007',] |>
ggplot(

  # Choose aesthetic mapping parameters
  aes(x = Longitude, y = Latitude, colour = as.factor(Family_FS))) +

  # Choose to use a scatter plot, with shape parameters
  #geom_point(size = 3) +

  # Choose to use a scatter plot, with size parameters.
  # Include a slight jitter so that all individual points can be seen.
  # Set a random seed so that the jitter is reproducible
  geom_point(position = position_jitter(width = 0.001,
                                         height = 0.001, seed = 39), size = 2) +

  theme_classic() +

  # Break the x axis
  scale_x_break(breaks = c(0.0176511, 0.3166511), scales = 3) +

  # Choose x axis ticks to show, this somehow also duplicates it
  scale_x_continuous(breaks = c(-0.0033489, 0.0166511, 0.3266511, 0.3466511,
                                0.3666511)) +

  # Duplicate y axis on the right
  scale_y_continuous(sec.axis = dup_axis()) +

  scale_colour_manual(values = c("dodgerblue2", "#E31A1C", "green4",
                                "#6A3D9A", "#FF7F00", "black", "gold1",
                                "skyblue2", "maroon3", "#FDBF6F", "palegreen2",
                                "burlywood4", "orchid1", "darkturquoise",

```

```

        "darkorange4"),
na.value = alpha('grey', 0.5)) +

# Set a transparent colour for individuals that are NA
#scale_alpha_manual(values = rep(1, times = 15), na.value = 0.5) +

# Edit the legend
labs(colour = 'Fullsib Family') +
guides(colour = FALSE) +

theme(
  # Remove axis tick labels for confidentiality
  axis.text = element_blank(),

  # Make axis titles bold
  axis.title = element_text(face = 2)) #
map3

# Scale:
# X axis ticks are 0.02 degrees longitude.
# Y axis ticks are 0.01 degrees latitude.
# Thus, X axis marks are approximately (very close to) 2 km apart.
# Y axis marks are approximately 1.11 km apart.

Figure5a <- NanguraSpinosa.dat[NanguraSpinosa.dat$Year != '2006-2007',] |>
ggplot(

  # Choose aesthetic mapping parameters
  aes(x = Longitude, y = Latitude, colour = as.factor(Family_HS))) +

  # Choose to use a scatter plot, with shape parameters
  #geom_point(size = 3) +

  # Choose to use a scatter plot, with size parameters.
  # Include a slight jitter so that all individual points can be seen.
  # Set a random seed so that the jitter is reproducible
  geom_point(position = position_jitter(width = 0.001,
                                         height = 0.001, seed = 39), size = 2) +

  theme_classic() +

  # Break the x axis
  scale_x_break(breaks = c(0.0176511, 0.3166511), scales = 3) +

  # Choose x axis ticks to show, this somehow also duplicates it
  scale_x_continuous(
    breaks = c(-0.0033489, 0.0166511, 0.3266511, 0.3466511, 0.3666511)) +

```

```

# Duplicate y axis on the right
scale_y_continuous(sec.axis = dup_axis()) +

scale_colour_manual(values = c("dodgerblue2", "#E31A1C", "green4",
                                "#6A3D9A", "#FF7F00", "black", "gold1",
                                "skyblue2", "maroon3", "#FDBF6F",
                                "palegreen2", "burlywood4", "orchid1",
                                "darkturquoise", "darkorange4"),
                    na.value = alpha('grey', 0.5)) +

# Set a transparent colour for individuals that are NA
#scale_alpha_manual(values = rep(1, times = 15), na.value = 0.5) +

# Edit the legend
labs(colour = 'Halfsib Family') +
guides(colour = FALSE) +
theme(
  # Remove axis tick labels for confidentiality
  axis.text = element_blank(),

  # Make axis titles bold
  axis.title = element_text(face = 2)) +

# Label geographic locations
annotate("text", x = 0.008 , y = 0.00166043,
        label = 'Nangur', fontface = 2, size = 5) +
annotate("text", x = 0.3366511, y = -0.01833957,
        label = 'Oakview', fontface = 2, size = 5)

```

Figure5a

```

# Scale:
# X axis ticks are 0.02 degrees longitude.
# Y axis ticks are 0.01 degrees latitude.
# Thus, X axis marks are approximately (very close to) 2 km apart.
# Y axis marks are approximately 1.11 km apart.

```

Halfsibs

Reference Map

Extract an Australian shape file with state and coastline information

```
sf_oz_states <- ozmap_data("states") # states and coastlines
```

Use sf package to read in national park kml's. KML files come from Queensland Government's Wetland Info database (this will be cited in the main manuscript).

```

# Read in
nangurNP <- st_read(dsn = "../data/mapping/national-park-nangur.kml")

# Read in
oakviewNP <- st_read(dsn = "../data/mapping/national-park-oakview.kml")

# Bind together
spinosaNK <- rbind(nangurNP, oakviewNP)

```

Plot Australia-wide Map

```

map.5a <- ggplot() +
  geom_sf(data = sf_oz_states[3,]) + # Plot shape file of Queensland
  geom_sf(data = sf_oz_states, alpha = 0) + # Map the rest of Australia
  theme_classic() + # Clean background w/o grid
  theme(axis.title = element_blank(), # The following removes axes features.
        axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())
map.5a

```

Plot Queensland wide map: national park kml's with ggplot, sf, and finally add an ozmap on top

```

map.5b <- ggplot() +
  labs(x="Longitude", y="Latitude") + # Axis titles
  geom_sf(data = sf_oz_states[3,]) + # Plot shape file of Queensland
  geom_sf(data = sf_oz_states, alpha = 0) + # Map the rest of Australia
  #geom_sf(data = nangurNP, fill = 'forestgreen') + # Plot sf of Nangur NP
  #geom_sf(data = oakviewNP, fill = 'royalblue') + # Plot sf of Oakview NP
  theme_bw() + # Not my favourite theme, but good for maps!
  ylim(c(-30, -9)) + # Set the limits so the focus is on QLD (Lats)
  xlim(c(137, 155)) + # Set the limits so the focus is on QLD (Longs)
  geom_text(aes(y = -20.917574, x = 142.702789), # Annotate the State name
            label = 'QUEENSLAND',
            size = 4.5,
            color = 'black',
            face = 2) +
  annotate('rect', # Annotate a box for the next map
         ymin = -26.4, ymax = -25.8,
         xmin = 151.96, xmax = 152.7,
         colour = 'black',
         alpha = 0,
         size = 1
        ) +
  geom_point(aes(y = -27.4705, x = 153.0260), # Plot a point for Brisbane
            colour = 'black',
            size = 2) +
  geom_text(aes(y = -27.4705, x = 151.1), # Label Brisbane
            label = 'Brisbane',
            face = 2) +
  geom_point(aes(y = -19.2743, x = 146.6514), # Plot a point for Townsville
            colour = 'black',
            size = 2) +

```

```

geom_text(aes(y = -18.8, x = 148.8), # Label Townsville
          label = 'Townsville',
          face = 2) +
geom_point(aes(y = -16.9203, x = 145.7710), # Plot a point for Cairns
           colour = 'black',
           size = 2) +
geom_text(aes(y = -16.9203, x = 147.2), # Label Cairns
          label = 'Cairns',
          face = 2)

```

map.5b

Plot National Park Map

```

# Plot map
map.5c <- ggplot() +
  labs(x="Longitude", y="Latitude") + # Axis titles
  geom_sf(data = nangurNP, fill = 'forestgreen') + # Plot sf of Nangur NP
  geom_sf(data = oakviewNP, fill = 'royalblue') + # Plot sf of Oakview NP
  theme_bw() + # Not my favourite theme, but good for maps!
  ylim(c(-26.4, -25.8)) + # Latitude limits
  xlim(c(151.96, 152.7)) + # Longitude limits
  geom_point(aes(y = -26.1836, x = 152.6624), # Plot a point for Gympie
             colour = 'black',
             size = 3) +
  geom_text(aes(y = -26.21, x = 152.6624), # Label Gympie
            label = 'Gympie',
            face = 2) +
  geom_text(aes(y = -26.16, x = 152), # Label Nangur NP
            label = 'Nangur NP',
            face = 2) +
  geom_text(aes(y = -26.1, x = 152.33), # Label Nangur NP
            label = 'Oakview NP',
            face = 2)

```

map.5c

```

leafletNP <- leaflet(spinosaNP,
                    options = leafletOptions(zoomControl = FALSE)) %>%
  addTiles() %>%
  addPolygons(weight = 3,
              opacity = 1,
              fillOpacity = 0,
              color = c('forestgreen', 'royalblue')) %>%
  setView( lng = 152.3, lat = -26.15, zoom = 10 ) %>%
  addProviderTiles("Esri.WorldImagery") %>%
  addScaleBar(position = 'topright',
              options = scaleBarOptions(imperial = F)) %>%
  addGraticule(interval = 0.1, style = list(color = "#000000", weight = 0.5) )

```

References

This reference list only includes references and packages that are cited in this R Markdown document. For a full reference list, please see the main text.

- Catchen J, Hohenlohe PA, Bassham S, et al (2013) Stacks: an analysis tool set for population genomics. *Mol Ecol* 22:3124–3140. <https://doi.org/10.1111/mec.12354>
- Catchen JM, Amores A, Hohenlohe P, et al (2011) Stacks: Building and Genotyping Loci De Novo From Short-Read Sequences. *G3 Genes|Genomes|Genetics* 1:. <https://doi.org/10.1534/g3.111.000240>
- Chang CC, Chow CC, Tellier LC, et al (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets. *Gigascience* 4:7. <https://doi.org/10.1186/s13742-015-0047-8>
- Danecek P, Auton A, Abecasis G, et al (2011) The variant call format and VCFtools. *Bioinformatics* 27:. <https://doi.org/10.1093/bioinformatics/btr330>
- Do C, Waples RS, Peel D, et al (2014) NeEstimator v2: re-implementation of software for the estimation of contemporary effective population size (N_e) from genetic data. *Mol Ecol Resour* 14:209–214. <https://doi.org/10.1111/1755-0998.12157>
- Harrison KA, Magrath MJL, Yen JDL, et al (2019) Lifetime Fitness Costs of Inbreeding and Being Inbred in a Critically Endangered Bird. *Current Biology* 29:2711-2717.e4. <https://doi.org/10.1016/j.cub.2019.06.064>
- Schmidt TL, Jasper M, Weeks AR, Hoffmann AA (2021) Unbiased population heterozygosity estimates from genome-wide sequence data. *Methods Ecol Evol* 12:1888–1898. <https://doi.org/10.1111/2041-210X.13659>
- Wang J (2014) Marker-based estimates of relatedness and inbreeding coefficients: an assessment of current methods. *J Evol Biol* 27:518–530. <https://doi.org/10.1111/jeb.12315>
- Waples RS (2015) Testing for Hardy–Weinberg Proportions: Have We Lost the Plot? *Journal of Heredity* 106:1–19. <https://doi.org/10.1093/jhered/esu062>
- Waples RS, Anderson EC (2017) Purging putative siblings from population genetic data sets: a cautionary view. *Mol Ecol* 26:1211–1224. <https://doi.org/10.1111/mec.14022>
- Weeks AR, Sgro CM, Young AG, et al (2011) Assessing the benefits and risks of translocations in changing environments: a genetic perspective. *Evol Appl* 4:709–725. <https://doi.org/10.1111/j.1752-4571.2011.00192.x>

Packages

```
citation('devtools')
```

```
## To cite package 'devtools' in publications use:
##
## Wickham H, Hester J, Chang W, Bryan J (2022). _devtools: Tools to
## Make Developing R Packages Easier_.
## doi:10.32614/CRAN.package.devtools
## <https://doi.org/10.32614/CRAN.package.devtools>, R package version
## 2.4.5, <https://CRAN.R-project.org/package=devtools>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
```

```
## title = {devtools: Tools to Make Developing R Packages Easier},
## author = {Hadley Wickham and Jim Hester and Winston Chang and Jennifer Bryan},
## year = {2022},
## note = {R package version 2.4.5},
## url = {https://CRAN.R-project.org/package=devtools},
## doi = {10.32614/CRAN.package.devtools},
## }
```

```
citation('tidyverse')
```

```
## To cite package 'tidyverse' in publications use:
```

```
##
```

```
## Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
## Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller
## E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V,
## Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to
## the tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
## doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.
```

```
##
```

```
## A BibTeX entry for LaTeX users is
```

```
##
```

```
## @Article{,
## title = {Welcome to the {tidyverse}},
## author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy D'Agostini
## year = {2019},
## journal = {Journal of Open Source Software},
## volume = {4},
## number = {43},
## pages = {1686},
## doi = {10.21105/joss.01686},
## }
```

```
citation('knitr')
```

```
## To cite package 'knitr' in publications use:
```

```
##
```

```
## Xie Y (2025). _knitr: A General-Purpose Package for Dynamic Report
## Generation in R_. R package version 1.50, <https://yihui.org/knitr/>.
```

```
##
```

```
## Yihui Xie (2015) Dynamic Documents with R and knitr. 2nd edition.
## Chapman and Hall/CRC. ISBN 978-1498716963
```

```
##
```

```
## Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible
## Research in R. In Victoria Stodden, Friedrich Leisch and Roger D.
## Peng, editors, Implementing Reproducible Computational Research.
## Chapman and Hall/CRC. ISBN 978-1466561595
```

```
##
```

```
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('RColorBrewer')
```

```
## To cite package 'RColorBrewer' in publications use:
##
##   Neuwirth E (2022). _RColorBrewer: ColorBrewer Palettes_.
##   doi:10.32614/CRAN.package.RColorBrewer
##   <https://doi.org/10.32614/CRAN.package.RColorBrewer>, R package
##   version 1.1-3, <https://CRAN.R-project.org/package=RColorBrewer>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {RColorBrewer: ColorBrewer Palettes},
##     author = {Erich Neuwirth},
##     year = {2022},
##     note = {R package version 1.1-3},
##     url = {https://CRAN.R-project.org/package=RColorBrewer},
##     doi = {10.32614/CRAN.package.RColorBrewer},
##   }
```

```
citation('vcfR')
```

```
## To cite vcfR in publications use: To cite vcfR in publications use:
##
##   Knaus BJ, Grünwald NJ (2017). "VCFR: a package to manipulate and
##   visualize variant call format data in R." _Molecular Ecology
##   Resources_, *17*(1), 44-53. ISSN 757,
##   <https://dx.doi.org/10.1111/1755-0998.12549>.
##
##   Knaus BJ, Grünwald NJ (2016). "VcfR: an R package to manipulate and
##   visualize VCF format data." _BioRxiv_.
##   <https://dx.doi.org/10.1101/041277>.
##
## We have invested a lot of time and effort in creating vcfR, please cite
## it when using it for data analysis. We have invested a lot of time and
## effort in creating vcfR, please cite it when using it for data
## analysis.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('adegenet')
```

```
## To cite the adegenet package:
##
##   Jombart, T. (2008) adegenet: a R package for the multivariate
##   analysis of genetic markers. Bioinformatics 24: 1403-1405. doi:
##   10.1093/bioinformatics/btn129
##
##   Jombart T. and Ahmed I. (2011) adegenet 1.3-1: new tools for the
##   analysis of genome-wide SNP data. Bioinformatics. doi:
```

```
## 10.1093/bioinformatics/btr521
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('igraph')
```

```
## To cite 'igraph' in publications use:
##
## Csardi G, Nepusz T (2006). "The igraph software package for complex
## network research." _InterJournal_, *Complex Systems*, 1695.
## <https://igraph.org>.
##
## Csárdi G, Nepusz T, Traag V, Horvát Sz, Zanini F, Noom D, Müller K
## (2026). _igraph: Network Analysis and Visualization in R_.
## doi:10.5281/zenodo.7682609 <https://doi.org/10.5281/zenodo.7682609>, R
## package version 2.1.4, <https://CRAN.R-project.org/package=igraph>.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
#citation('ggtern')
citation('reshape2')
```

```
## To cite reshape2 in publications use:
##
## Hadley Wickham (2007). Reshaping Data with the reshape Package.
## Journal of Statistical Software, 21(12), 1-20. URL
## http://www.jstatsoft.org/v21/i12/.
##
## A BibTeX entry for LaTeX users is
##
## @Article{,
##   title = {Reshaping Data with the {reshape} Package},
##   author = {Hadley Wickham},
##   journal = {Journal of Statistical Software},
##   year = {2007},
##   volume = {21},
##   number = {12},
##   pages = {1--20},
##   url = {http://www.jstatsoft.org/v21/i12/},
## }
```

```
citation('geodist')
```

```
## To cite package 'geodist' in publications use:
##
## Padgham M, Sumner M (2025). _geodist: Fast, Dependency-Free Geodesic
## Distance Calculations_. doi:10.32614/CRAN.package.geodist
## <https://doi.org/10.32614/CRAN.package.geodist>, R package version
```

```
## 0.1.1, <https://CRAN.R-project.org/package=geodist>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {geodist: Fast, Dependency-Free Geodesic Distance Calculations},
##   author = {Mark Padgham and Michael D. Sumner},
##   year = {2025},
##   note = {R package version 0.1.1},
##   url = {https://CRAN.R-project.org/package=geodist},
##   doi = {10.32614/CRAN.package.geodist},
## }
```

```
citation('ggridges')
```

```
## To cite package 'ggridges' in publications use:
##
## Wilke C (2024). _ggridges: Ridgeline Plots in 'ggplot2'_.
## doi:10.32614/CRAN.package.ggridges
## <https://doi.org/10.32614/CRAN.package.ggridges>, R package version
## 0.5.6, <https://CRAN.R-project.org/package=ggridges>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {ggridges: Ridgeline Plots in 'ggplot2'},
##   author = {Claus O. Wilke},
##   year = {2024},
##   note = {R package version 0.5.6},
##   url = {https://CRAN.R-project.org/package=ggridges},
##   doi = {10.32614/CRAN.package.ggridges},
## }
```

```
citation('viridis')
```

```
## To cite viridis/viridisLite in publications use:
##
## Simon Garnier, Noam Ross, Robert Rudis, Antônio P. Camargo, Marco
## Sciaini, and Cédric Scherer (2024). viridis(Lite) -
## Colorblind-Friendly Color Maps for R. viridis package version 0.6.5.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {{viridis(Lite)} - Colorblind-Friendly Color Maps for R},
##   author = {{Garnier} and {Simon} and {Ross} and {Noam} and {Rudis} and {Robert} and {Camargo} and
##   year = {2024},
##   note = {viridis package version 0.6.5},
##   url = {https://sjmgarnier.github.io/viridis/},
##   doi = {10.5281/zenodo.4679423},
## }
```

```
citation('poppr')
```

```
## To cite poppr in publications or presentations, please specify poppr
## version 2.9.7 and with the following citation:
##
## Kamvar ZN, Tabima JF, Grünwald NJ. (2014) Poppr: an R package for
## genetic analysis of populations with clonal, partially clonal, and/or
## sexual reproduction. PeerJ 2:e281. doi: 10.7717/peerj.281
##
## Kamvar ZN, Brooks JC and Grünwald NJ (2015) Novel R tools for
## analysis of genome-wide population genetic data with emphasis on
## clonality. Front. Genet. 6:208. doi: 10.3389/fgene.2015.00208
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('patchwork')
```

```
## To cite package 'patchwork' in publications use:
##
## Pedersen T (2025). _patchwork: The Composer of Plots_.
## doi:10.32614/CRAN.package.patchwork
## <https://doi.org/10.32614/CRAN.package.patchwork>, R package version
## 1.3.1, <https://CRAN.R-project.org/package=patchwork>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {patchwork: The Composer of Plots},
##   author = {Thomas Lin Pedersen},
##   year = {2025},
##   note = {R package version 1.3.1},
##   url = {https://CRAN.R-project.org/package=patchwork},
##   doi = {10.32614/CRAN.package.patchwork},
## }
```

```
citation('ggbreak')
```

```
## To cite ggbreak in publications use:
##
## S Xu, M Chen, T Feng, L Zhan, L Zhou, G Yu. Use ggbreak to
## effectively utilize plotting space to deal with large datasets and
## outliers. Frontiers in Genetics. 2021, 12:774846. doi:
## 10.3389/fgene.2021.774846
##
## A BibTeX entry for LaTeX users is
##
## @Article{,
##   title = {Use ggbreak to effectively utilize plotting space to deal with large datasets and outli
##   author = {{Shuangbin Xu} and Meijun Chen and Tingze Feng and Li Zhan and Lang Zhou and Guangchuan
```

```
##   year = {2021},
##   journal = {Frontiers in Genetics},
##   volume = {12},
##   pages = {774846},
##   doi = {10.3389/fgene.2021.774846},
## }
```

```
citation('HardyWeinberg')
```

```
## To cite HardyWeinberg in publications use:
##
##   Graffelman J (2015). "Exploring Diallelic Genetic Markers: The
##   HardyWeinberg Package." _Journal of Statistical Software_, *64*(3),
##   1-23. doi:10.18637/jss.v064.i03
##   <https://doi.org/10.18637/jss.v064.i03>.
##
##   Graffelman J, Morales-Camarena J (2008). "Graphical tests for
##   Hardy-Weinberg Equilibrium based on the ternary plot." _Human
##   Heredity_, *65*(2), 77-84. doi:10.1159/000108939
##   <https://doi.org/10.1159/000108939>.
##
##   Graffelman J, Weir BS (2016). "Testing for Hardy-Weinberg equilibrium
##   at bi-allelic genetic markers on the X chromosome." _Heredity_,
##   *116*(6), 558-568. doi:10.1038/hdy.2016.20
##   <https://doi.org/10.1038/hdy.2016.20>.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('netviewr')
```

```
## To cite package 'netviewr' in publications use:
##
##   Steinig E (2024). _netviewr: Network Visualization And Population
##   Structure_. R package version 2.1.0.9000, commit
##   b48b9795678eeeea1bafc83b52acdb29a6084dab9,
##   <https://github.com/esteinig/netviewr>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {netviewr: Network Visualization And Population Structure},
##     author = {Eike Steinig},
##     year = {2024},
##     note = {R package version 2.1.0.9000, commit b48b9795678eeeea1bafc83b52acdb29a6084dab9},
##     url = {https://github.com/esteinig/netviewr},
##   }
##
## ATTENTION: This citation information has been auto-generated from the
## package DESCRIPTION file and may need manual editing, see
## 'help("citation")'.
```

```
citation('tinytex')
```

```
## To cite package 'tinytex' in publications use:
##
## Xie Y (2025). _tinytex: Helper Functions to Install and Maintain TeX
## Live, and Compile LaTeX Documents_. R package version 0.57,
## <https://github.com/rstudio/tinytex>.
##
## Xie Y (2019). "TinyTeX: A lightweight, cross-platform, and
## easy-to-maintain LaTeX distribution based on TeX Live." _TUGboat_,
## *40*(1), 30-32. <https://tug.org/TUGboat/Contents/contents40-1.html>.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('ozmaps')
```

```
## To cite package 'ozmaps' in publications use:
##
## Sumner M (2021). _ozmaps: Australia Maps_.
## doi:10.32614/CRAN.package.ozmaps
## <https://doi.org/10.32614/CRAN.package.ozmaps>, R package version
## 0.4.5, <https://CRAN.R-project.org/package=ozmaps>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {ozmaps: Australia Maps},
##   author = {Michael Sumner},
##   year = {2021},
##   note = {R package version 0.4.5},
##   url = {https://CRAN.R-project.org/package=ozmaps},
##   doi = {10.32614/CRAN.package.ozmaps},
## }
```

```
citation('dartR')
```

```
## To cite dartR in publications use:
##
## Gruber, B, Unmack, PJ, Berry, OF, Georges, A. (2018). dartr: An r
## package to facilitate analysis of SNP data generated from reduced
## representation genome sequencing. Molecular Ecology Resources 18:
## 691-699. https://doi.org/10.1111/1755-0998.12745
##
## Mijangos, J. L., Gruber, B., Berry, O., Pacioni, C., & Georges, A.
## (2022). dartR v2: An accessible genetic analysis platform for
## conservation, ecology and agriculture. Methods in Ecology and
## Evolution, (early access). https://doi.org/10.1111/2041-210X.13918
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('sf')
```

```
## To cite package sf in publications, please use:
##
##   Pebesma, E., & Bivand, R. (2023). Spatial Data Science: With
##   Applications in R. Chapman and Hall/CRC.
##   https://doi.org/10.1201/9780429459016
##
##   Pebesma, E., 2018. Simple Features for R: Standardized Support for
##   Spatial Vector Data. The R Journal 10 (1), 439-446,
##   https://doi.org/10.32614/RJ-2018-009
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex()', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('ozmaps')
```

```
## To cite package 'ozmaps' in publications use:
##
##   Sumner M (2021). _ozmaps: Australia Maps_.
##   doi:10.32614/CRAN.package.ozmaps
##   <https://doi.org/10.32614/CRAN.package.ozmaps>, R package version
##   0.4.5, <https://CRAN.R-project.org/package=ozmaps>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {ozmaps: Australia Maps},
##     author = {Michael Sumner},
##     year = {2021},
##     note = {R package version 0.4.5},
##     url = {https://CRAN.R-project.org/package=ozmaps},
##     doi = {10.32614/CRAN.package.ozmaps},
##   }
```

```
citation('leaflet')
```

```
## To cite package 'leaflet' in publications use:
##
##   Cheng J, Schloerke B, Karambelkar B, Xie Y, Aden-Buie G (2025).
##   _leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet'
##   Library_. doi:10.32614/CRAN.package.leaflet
##   <https://doi.org/10.32614/CRAN.package.leaflet>, R package version
##   2.2.3, <https://CRAN.R-project.org/package=leaflet>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet'
## Library},
##     author = {Joe Cheng and Barret Schloerke and Bhaskar Karambelkar and Yihui Xie and Garrick Aden-1
```

```
##   year = {2025},
##   note = {R package version 2.2.3},
##   url = {https://CRAN.R-project.org/package=leaflet},
##   doi = {10.32614/CRAN.package.leaflet},
## }
```

```
citation('splitstackshape')
```

```
## To cite package 'splitstackshape' in publications use:
##
##   Mahto A (2026). _splitstackshape: Stack and Reshape Datasets After
##   Splitting Concatenated Values_.
##   doi:10.32614/CRAN.package.splitstackshape
##   <https://doi.org/10.32614/CRAN.package.splitstackshape>, R package
##   version 1.4.8.1,
##   <https://CRAN.R-project.org/package=splitstackshape>.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {splitstackshape: Stack and Reshape Datasets After Splitting Concatenated Values},
##     author = {Ananda Mahto},
##     year = {2026},
##     note = {R package version 1.4.8.1},
##     url = {https://CRAN.R-project.org/package=splitstackshape},
##     doi = {10.32614/CRAN.package.splitstackshape},
##   }
```

Markdown Appendix 1: How files for Netview were generated

VCF to mdist for NetviewR.

In this example, there were four vcf files output with different subsets of loci and individuals, for which mdist files were desired. The vcf files have been named Alpha, Beta, Gamma, and Delta.

Convert to Plink First I need to convert the four output vcf files from VCF to Plink 1.9 (Chang et al., 2015) format:

```
vcftools --vcf ../../1_vcf_filtering/NS_M2n2m3_ChildAlpha.recode.vcf --plink --out
NS_M2n2m3_ChildAlpha
```

```
vcftools --vcf ../../1_vcf_filtering/NS_M2n2m3_ChildBeta.recode.vcf --plink --out NS_M2n2m3_ChildBeta
```

```
vcftools --vcf ../../1_vcf_filtering/NS_M2n2m3_ChildGamma.recode.vcf --plink --out
NS_M2n2m3_ChildGamma
```

```
vcftools --vcf ../../1_vcf_filtering/NS_M2n2m3_ChildDelta.recode.vcf --plink --out
NS_M2n2m3_ChildDelta
```

An issue with this conversion is that Plink 1.9 doesn't like the .map file to contain locus numbers. So I have to correct the file so that it can read it, by prepending the word contig to each of the locus numbers. AND down the tack I need to include the code `-allow-extra-chr` to permit the reading of these.

I can make the correction in R.

```

ChildList <- c('ChildAlpha', 'ChildBeta', 'ChildGamma', 'ChildDelta')
# A list of the names of each file

for (i in 1:4){
  # A for loop with 4 iterations

  MapPath <- paste('NS_M2n2m3_', ChildList[i], '.map', sep = "")
  # The path to the file I am modifying. Allowing it to iterate through
  # different file names

  map <- read.delim(file = MapPath, sep = "\t", header = F)
  # read in plink file.

  map$V1 = paste("contig", map$V1, sep = "")
  # Add 'contig' as text to every row in the first column so that
  # it does not read as a chromosome number in plink

  #write.table(map, file = MapPath, sep = "\t", dec = ".", row.names = FALSE,
  # col.names = FALSE, quote = FALSE)
# Export as a table that looks identical to the one read in.
  # NOTE I HAVE DISABLED THIS LINE [commented out] SO THAT THE FILES ARE NOT
  # OVERWRITTEN IF THIS CHUNK GETS RUN

}

```

The .map files were then uploaded to the hpc and overwrote the previous versions of the file.

Generate distance matrix

I can now use Plink 1.9 (Chang et al., 2015) to generate a square (symmetric) distance (1-IBS) matrix for each of the child datasets.

I then used the following code in plink 1.9 to generate square (symmetric) distance (1-IBS) matrices using this code.

```

plink1.9 --file NS_M2n2m3_ChildAlpha --distance square 1-ibs --allow-extra-chr --out
dist_ChildAlpha

plink1.9 --file NS_M2n2m3_ChildBeta --distance square 1-ibs --allow-extra-chr --out
dist_ChildBeta

plink1.9 --file NS_M2n2m3_ChildGamma --distance square 1-ibs --allow-extra-chr --out
dist_ChildGamma

plink1.9 --file NS_M2n2m3_ChildDelta --distance square 1-ibs --allow-extra-chr --out
dist_ChildDelta

```

The distance matrices are stored in .mdist and their labels in .mdist.id files.

Markdown Appendix 2: How genotypes were converted to Colony format

Note: DartRverse has functions that allow running Colony from R which is much more efficient. Also, many of the steps involving excel really can (and maybe should) be done using R.

```
#####
# COLconversion
# Nic Bail
# 12/08/2021
#####

#### Packages ####
library(tidyverse)

library(splitstackshape)

#### Intent ####
#' This R script is designed to take a .csv file and create another .csv file.
#' The final .csv file will be useful for making a .txt file for the software Colony to use.
#'
#' The first csv file has cells filled with one of the following:
#' 0/0 0/1 1/1 or ./
#' I want each column to be split into two.
#' Giving cells that look like this:
#' 1 1 1 2 2 2 0 0
#' i.e. I want .s to be 0s, 0s to be 1s, and 1s to be 2s.

#### Prior Steps ####
# I opened my .vcf file in Excel.
# I copied just the matrix of individual data.
# I transposed this data so that individuals were the rows.
# I kept just the first three characters - the genotype information, of each cell.
# I named the first column individual. All others were kept blank.
# Individuals were named Ind1 - Ind483
# I then saved this as a .csv file to upload to R.

#### R steps ####

INPUT.CSV = read.csv(file = 'input/NS_M2n2m3_ChildDelta_cleanNan/NS_M2n2m3_ChildDelta_cleanNan_Colin.csv')
# Read file.

STEP_1.csv = INPUT.CSV
# Copy the file just in case I want to modify it, so the original is still accessible.

STEP_2.CSV = cSplit(STEP_1.csv, splitCols = c(2:ncol(STEP_1.csv)), sep="/", direction = "wide", drop = FALSE)
# Separate every column into two columns using the / as a delimiter.
# The data frame doubles in width.

STEP_3.CSV = sapply(STEP_2.CSV[,2:ncol(STEP_2.CSV)], gsub, pattern = "1", replacement = "2", fixed = TRUE)
# Make all the ones into twos.
# This step must be first.
# Individual names are lost in this step!!

STEP_3.CSV = as.data.frame(STEP_3.CSV)
# Make into a dataframe (necessary for step 4 to work)
```

```

STEP_4.CSV = sapply(STEP_3.CSV, gsub, pattern = "0", replacement = "1", fixed = TRUE, simplify = TRUE)
  # make all the zeros into ones.
  # this step must be second so it does not overwrite and is not overwritten.

STEP_4.CSV = as.data.frame(STEP_4.CSV)

STEP_5.CSV = sapply(STEP_4.CSV, gsub, pattern = ".", replacement = "0", fixed = TRUE, simplify = TRUE)
  # make all the dots into zeros.
  # the last step

STEP_5.CSV = as.data.frame(STEP_5.CSV)
  # Make it a more useful dataframe.

STEP_6.CSV = cbind(STEP_2.CSV[,1], STEP_5.CSV)
  # This re-adds the individual names as the first column.

  # This is ready for export.

write.table(STEP_6.CSV, "File_name_ColonyInput.txt", append = FALSE, sep = "\t", dec = ".", row.names =
  # exported as a .txt file

```