NB Population Genomic Analysis on Haplotig Masked Neutral SNPs

Amy Zyck

2/24/2021

Population analyses on Neutral SNP dataset (Haplotig Masked Genome)

I first created a new directory `NeutralHap` within `PATH: /home/azyck/NB_capture/NB_ddhaplo/NB_PopGenHap`. I linked `neutrallocihap.recode.vcf` to this working directory. The RMarkdown file should be saved in this same working directory.

```
#Loading all the necessary packages
library(adegenet)
```

```
## Loading required package: ade4
```

```
## Registered S3 method overwritten by 'spdep':
##   method   from
##   plot.mst ape
```

```
##
##    /// adegenet 2.1.3 is loaded ////////////
##
##    > overview: '?adegenet'
##    > tutorials/doc/questions: 'adegenetWeb()'
##    > bug reports/feature requests: adegenetIssues()
```

```
library(vcfR)
```

```
##
##    *****       ***   vcfR   ***       *****
##    This is vcfR 1.12.0
##      browseVignettes('vcfR') # Documentation
##      citation('vcfR') # Citation
##    *****       *****      *****       *****
```

```
library("radiator") # Conversion from vcf to a lot of other formats
```

```
## ******************************** IMPORTANT NOTICE ********************************
## radiator v.1.0.0 was modified heavily.
## Read functions documentation and available vignettes.
##
## For reproducibility:
##     radiator version: 1.0.0
##     radiator build date: R 3.5.1; ; 2019-03-20 13:31:04 UTC; unix
##     Keep zenodo DOI.
```

```
## ********************************************************************************
library("dplyr")

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library("hierfstat")

##
## Attaching package: 'hierfstat'

## The following object is masked from 'package:adegenet':
##
##     read.fstat
library("ggplot2") #For plotting
library("reshape2") #For plotting
library("plyr")

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
library("cowplot") #For plotting manuscript figs
library(PCAviz)  #Visualizing output of PCA
library("stringr")
library("bigsnpr") # package for Linkage Disequilibrium pruning

## Loading required package: bigstatsr
```

All Neutral SNPs (Haplotig Masked Genome)

Making files

Make genind object

`neutrallocihap.recode.vcf` contains neutral SNPs for populations PVD, GB, BIS, and NAR identified following read alignment to Haplotig masked genome. Steps for generating this VCF file are located in

EecSeq_Cvirginica_dDocent.md and EecSeq_Cvirginica_Filtering.md. Population NIN was removed from the VCF file following steps in EecSeq_Cvirginica_OutlierDetection.md.

`strata` contains population, environmental, and library information for each sample - can be accessed here.

```
my_vcf <- read.vcfR("neutrallocihap.recode.vcf")
```

```
## Scanning file to determine attributes.
## File attributes:
##   meta lines: 77
##   header_line: 78
##   variant count: 90861
##   column count: 49
##
Meta line 77 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 90861
##   Character matrix gt cols: 49
##   skip: 0
##   nrows: 90861
##   row_num: 0
##
Processed variant 1000
Processed variant 2000
Processed variant 3000
Processed variant 4000
Processed variant 5000
Processed variant 6000
Processed variant 7000
Processed variant 8000
Processed variant 9000
Processed variant 10000
Processed variant 11000
Processed variant 12000
Processed variant 13000
Processed variant 14000
Processed variant 15000
Processed variant 16000
Processed variant 17000
Processed variant 18000
Processed variant 19000
Processed variant 20000
Processed variant 21000
Processed variant 22000
Processed variant 23000
Processed variant 24000
Processed variant 25000
Processed variant 26000
Processed variant 27000
Processed variant 28000
Processed variant 29000
Processed variant 30000
Processed variant 31000
```

```
Processed variant 32000
Processed variant 33000
Processed variant 34000
Processed variant 35000
Processed variant 36000
Processed variant 37000
Processed variant 38000
Processed variant 39000
Processed variant 40000
Processed variant 41000
Processed variant 42000
Processed variant 43000
Processed variant 44000
Processed variant 45000
Processed variant 46000
Processed variant 47000
Processed variant 48000
Processed variant 49000
Processed variant 50000
Processed variant 51000
Processed variant 52000
Processed variant 53000
Processed variant 54000
Processed variant 55000
Processed variant 56000
Processed variant 57000
Processed variant 58000
Processed variant 59000
Processed variant 60000
Processed variant 61000
Processed variant 62000
Processed variant 63000
Processed variant 64000
Processed variant 65000
Processed variant 66000
Processed variant 67000
Processed variant 68000
Processed variant 69000
Processed variant 70000
Processed variant 71000
Processed variant 72000
Processed variant 73000
Processed variant 74000
Processed variant 75000
Processed variant 76000
Processed variant 77000
Processed variant 78000
Processed variant 79000
Processed variant 80000
Processed variant 81000
Processed variant 82000
Processed variant 83000
Processed variant 84000
Processed variant 85000
```

```
Processed variant 86000
Processed variant 87000
Processed variant 88000
Processed variant 89000
Processed variant 90000
Processed variant: 90861
## All variants processed
```

```
strata <- read.table("strata", header=TRUE)
```

```
rad.filt <- vcfR2genind(my_vcf, strata = strata, pop = c(rep("BIS", 10),rep("GB", 10),rep("NAR", 10), re
```

```
rad.filt
```

```
## /// GENIND OBJECT /////////
##
##  // 40 individuals; 90,861 loci; 181,433 alleles; size: 82.8 Mb
##
##  // Basic content
##    @tab:  40 x 181433 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 1-2)
##    @loc.fac: locus factor for the 181433 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: adegenet::df2genind(X = t(x), sep = sep, pop = ..2, strata = ..1)
##
##  // Optional content
##    @pop: population of each individual (group size range: 10-10)
##    @strata: a data frame with 12 columns ( Individual, Population, Latitude, Longitude, Distance, SE
```

```
#Providing population names for plotting
pop_order <- c("BIS","GB","NAR","PVD")
```

Read in the other info from .strata file and extract information such as locality, latitude, and longitude.

```
info <- as.data.frame(read.table("strata",header = T,sep = "\t",stringsAsFactors = F))
mystrats <- as.data.frame(matrix(nrow = length(indNames(rad.filt)),ncol=10))
colnames(mystrats) <- c("Population","Latitude","Longitude","Distance","SE","Temperature","Salinity","pH
just.strats <- select(info,c("Population"))
stratted.filt <- strata(rad.filt, formula= Population, combine = TRUE,just.strats)
stratted.filt@other <- select(info, Latitude,Longitude,Distance,SE,Temperature,Salinity,pH,Chlorophylla
```

```
stratted.filt
```

```
## /// GENIND OBJECT /////////
##
##  // 40 individuals; 90,861 loci; 181,433 alleles; size: 82.8 Mb
##
##  // Basic content
##    @tab:  40 x 181433 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 1-2)
##    @loc.fac: locus factor for the 181433 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
```

```
##    @call: adegenet::df2genind(X = t(x), sep = sep, pop = ..2, strata = ..1)
##
##  // Optional content
##    @pop: population of each individual (group size range: 10-10)
##    @strata: a data frame with 1 columns ( Population )
##    @other: a list containing: Latitude  Longitude  Distance  SE  Temperature  Salinity  pH  Chlorophy
```

Repeat for quasi-independent set of SNPs

Steps for filtering are documented for OutFLANK in EecSeq_Cvirginica_OutlierDetection.md. However, I am repeating the steps here.

```
my_vcf <- read.vcfR("neutrallocihap.recode.vcf")
```

```
## Scanning file to determine attributes.
## File attributes:
##   meta lines: 77
##   header_line: 78
##   variant count: 90861
##   column count: 49
##
Meta line 77 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 90861
##   Character matrix gt cols: 49
##   skip: 0
##   nrows: 90861
##   row_num: 0
##
Processed variant 1000
Processed variant 2000
Processed variant 3000
Processed variant 4000
Processed variant 5000
Processed variant 6000
Processed variant 7000
Processed variant 8000
Processed variant 9000
Processed variant 10000
Processed variant 11000
Processed variant 12000
Processed variant 13000
Processed variant 14000
Processed variant 15000
Processed variant 16000
Processed variant 17000
Processed variant 18000
Processed variant 19000
Processed variant 20000
Processed variant 21000
Processed variant 22000
Processed variant 23000
Processed variant 24000
Processed variant 25000
```

```
Processed variant 26000
Processed variant 27000
Processed variant 28000
Processed variant 29000
Processed variant 30000
Processed variant 31000
Processed variant 32000
Processed variant 33000
Processed variant 34000
Processed variant 35000
Processed variant 36000
Processed variant 37000
Processed variant 38000
Processed variant 39000
Processed variant 40000
Processed variant 41000
Processed variant 42000
Processed variant 43000
Processed variant 44000
Processed variant 45000
Processed variant 46000
Processed variant 47000
Processed variant 48000
Processed variant 49000
Processed variant 50000
Processed variant 51000
Processed variant 52000
Processed variant 53000
Processed variant 54000
Processed variant 55000
Processed variant 56000
Processed variant 57000
Processed variant 58000
Processed variant 59000
Processed variant 60000
Processed variant 61000
Processed variant 62000
Processed variant 63000
Processed variant 64000
Processed variant 65000
Processed variant 66000
Processed variant 67000
Processed variant 68000
Processed variant 69000
Processed variant 70000
Processed variant 71000
Processed variant 72000
Processed variant 73000
Processed variant 74000
Processed variant 75000
Processed variant 76000
Processed variant 77000
Processed variant 78000
Processed variant 79000
```

```
Processed variant 80000
Processed variant 81000
Processed variant 82000
Processed variant 83000
Processed variant 84000
Processed variant 85000
Processed variant 86000
Processed variant 87000
Processed variant 88000
Processed variant 89000
Processed variant 90000
Processed variant: 90861
## All variants processed
```

```
geno <- extract.gt(my_vcf) # Character matrix containing the genotypes
position <- getPOS(my_vcf) # Positions in bp
chromosome <- getCHROM(my_vcf) # Chromosome information

G <- matrix(NA, nrow = nrow(geno), ncol = ncol(geno))

G[geno %in% c("0/0", "0|0")] <- 0
G[geno  %in% c("0/1", "1/0", "1|0", "0|1")] <- 1
G[geno %in% c("1/1", "1|1")] <- 2

# NA should be replaced with "9" to work with the functions in the OutFLANK package
G[is.na(G)] <- 9
```

Chromosomes need to be of class integer for this to work.

```
# Visualizing chromosomes
chrom_unique <- unique(chromosome)
print(chrom_unique)
```

```
##  [1] "NC_035789.1" "NC_035780.1" "NC_035781.1" "NC_035782.1" "NC_035783.1"
##  [6] "NC_035784.1" "NC_035785.1" "NC_035786.1" "NC_035787.1" "NC_035788.1"
```

```
# Removing "NC_" from the chromosome name so it can be converted to an integer
chrom_new <- chromosome%>%str_replace("NC_","")

# Converting the character string into an integer
chrom1 <- as.integer(chrom_new)
```

`chromosome` and `position` need to be sorted for imputation to work.

```
chrom_sort <- sort(chrom1)
pos_sort <- sort(position)
```

Note: This filtering program does not allow for missing genotype values.

I can either remove the missing genotype values or impute missing genotypes. I'm going to try both and see how the two methods differ during the trimming step.

Remove missing genotypes

```
# removing missing data
G_miss <- matrix(NA, nrow = nrow(geno), ncol = ncol(geno))
```

```
G_miss[geno %in% c("0/0", "0|0")] <- 0
G_miss[geno  %in% c("0/1", "1/0", "1|0", "0|1")] <- 1
G_miss[geno %in% c("1/1", "1|1")] <- 2

# removing missing data
G_miss[na.omit(G_miss)]
```

```
##     [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##    [37] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##    [73] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [109] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [145] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [181] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [217] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [253] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [289] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [325] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [361] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [397] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [433] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [469] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [505] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [541] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [577] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [613] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [649] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [685] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [721] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [757] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [793] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [829] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [865] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [901] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [937] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [973] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1009] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1045] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1081] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1117] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1153] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1189] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1225] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1261] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1333] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1369] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1405] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1441] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1477] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1513] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1549] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1585] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [1621] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
## [98857] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [98893] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [98929] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [98965] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99001] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99037] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99073] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99109] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99145] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99181] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99217] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99253] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99289] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99325] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99361] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99397] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99433] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99469] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99505] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99541] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99577] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99613] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99649] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99685] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99721] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99757] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99793] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99829] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99865] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99901] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99937] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [99973] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [ reached getOption("max.print") -- omitted 630779 entries ]
```

```
# Converting the genotype matrix to class FBM.code256 using the matrix where missing data was removed
G1_miss <- add_code256(big_copy(t(G_miss),type="raw"),code=bigsnpr:::CODE_012)
```

```
## Warning in replaceMat(x$address_rw, i, j, value): At least one value changed (nan -> 0)
##   while converting from R type 'double' to C type 'unsigned char (raw)'.
```

```
newpc_miss <- snp_autoSVD(G1_miss,infos.chr =chrom_sort,infos.pos = pos_sort)
```

```
##
## Phase of clumping (on MAF) at r^2 > 0.2.. keep 21253 SNPs.
## Discarding 6320 variants with MAC < 10.
##
## Iteration 1:
## Computing SVD..
## 0 outlier variant detected..
##
## Converged!
```

```
which_pruned_miss <- attr(newpc_miss, which="subset") # Indexes of remaining SNPS after pruning
length(which_pruned_miss)
```

```
## [1] 14933
```

Using `snp_fastImputeSimple` to impute missing genotypes

This requires the genotpe matrix to be converted to class FBM.code256.

> A reference class for storing and accessing up to 256 arbitrary different values using a Filebacked Big Matrix of type unsigned char.

```
# Converting the genotype matrix to class FBM.code256
G1 <- add_code256(big_copy(t(G),type="raw"),code=bigsnpr:::CODE_012)
```

The chunk below imputs missing genotypes in order for the following code to run. Fast imputation via mode, mean, sampling according to allele frequencies, or 0.

Method argument: Either "random" (sampling according to allele frequencies), "mean0" (rounded mean), "mean2" (rounded mean to 2 decimal places), "mode" (most frequent call).

Depending on the method I choose, the # of SNPs kept after running the following chunk changes, so I'm not sure which method to use.

```
# Imputing missing genotypes
G_missSimple <- snp_fastImputeSimple(G1,method = "mode") # Using mode or no method call results in the
```

Truncated SVD while limiting LD.

```
newpc<-snp_autoSVD(G_missSimple,infos.chr =chrom_sort,infos.pos = pos_sort)
```

```
##
## Phase of clumping (on MAF) at r^2 > 0.2.. keep 21225 SNPs.
## Discarding 6470 variants with MAC < 10.
##
## Iteration 1:
## Computing SVD..
## 39 outlier variants detected..
## 1 long-range LD region detected..
##
## Iteration 2:
## Computing SVD..
## 0 outlier variant detected..
##
## Converged!
```

```
which_pruned <- attr(newpc, which="subset") # Indexes of remaining SNPS after pruning
length(which_pruned)
```

```
## [1] 14716
```

Either removing missing data or imputation results in a similar number of trimmed SNPs, so I will continue forward with missing data removed.

```
invisible(lapply(which_pruned_miss, write, "pruned_data.txt", append=TRUE))
```

In terminal

```
$ mawk '!/#/' neutrallocihap.recode.vcf | cut -f1,2 > totalloci
$ NUM=(`cat totalloci | wc -l`)
$ paste <(seq 1 $NUM) totalloci > loci.plus.index
$ cat pruned_data.txt | parallel "grep -w ^{} loci.plus.index" | cut -f2,3> pruned_data_neutralhap.loci

$ head pruned_data_neutralhap.loci.txt
```

```
output:
NC_035789.1 148248
NC_035789.1 148254
NC_035789.1 243812
NC_035789.1 279683
NC_035789.1 279787
NC_035789.1 279807
NC_035789.1 285007
NC_035789.1 288608
NC_035789.1 377542
NC_035789.1 377835
```

Create VCF file with just the pruned_data loci

```
$ vcftools --vcf neutrallocihap.recode.vcf --recode --recode-INFO-all --positions pruned_data_neutralhap
```

```
output:
fter filtering, kept 40 out of 40 Individuals
Outputting VCF file...
After filtering, kept 14933 out of a possible 90861 Sites
Run Time = 4.00 seconds
```

```
my_vcf_u <- read.vcfR("pruned_data_neutralhap.recode.vcf")
```

```
## Scanning file to determine attributes.
## File attributes:
##   meta lines: 77
##   header_line: 78
##   variant count: 14933
##   column count: 49
##
Meta line 77 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 14933
##   Character matrix gt cols: 49
##   skip: 0
##   nrows: 14933
##   row_num: 0
##
Processed variant 1000
Processed variant 2000
Processed variant 3000
Processed variant 4000
Processed variant 5000
Processed variant 6000
Processed variant 7000
Processed variant 8000
Processed variant 9000
Processed variant 10000
Processed variant 11000
Processed variant 12000
Processed variant 13000
Processed variant 14000
Processed variant: 14933
```

```
## All variants processed
```

```
strata<- read.table("strata", header=TRUE)
```

```
rad.u <- vcfR2genind(my_vcf_u, strata = strata, pop = c(rep("BIS", 10),rep("GB", 10),rep("NAR", 10), rep
```

```
rad.u
```

```
## /// GENIND OBJECT /////////
##
##  // 40 individuals; 14,933 loci; 29,866 alleles; size: 13.6 Mb
##
##  // Basic content
##    @tab:  40 x 29866 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 2-2)
##    @loc.fac: locus factor for the 29866 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: adegenet::df2genind(X = t(x), sep = sep, pop = ..2, strata = ..1)
##
##  // Optional content
##    @pop: population of each individual (group size range: 10-10)
##    @strata: a data frame with 12 columns ( Individual, Population, Latitude, Longitude, Distance, SE
```

```
stratted.u <- strata(rad.u, formula= Population, combine = TRUE,just.strats)
stratted.u@other <- select(info, Latitude,Longitude, Distance,SE, Temperature,Salinity,pH,Chlorophylla,
```

```
stratted.u
```

```
## /// GENIND OBJECT /////////
##
##  // 40 individuals; 14,933 loci; 29,866 alleles; size: 13.6 Mb
##
##  // Basic content
##    @tab:  40 x 29866 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 2-2)
##    @loc.fac: locus factor for the 29866 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: adegenet::df2genind(X = t(x), sep = sep, pop = ..2, strata = ..1)
##
##  // Optional content
##    @pop: population of each individual (group size range: 10-10)
##    @strata: a data frame with 1 columns ( Population )
##    @other: a list containing: Latitude  Longitude  Distance  SE  Temperature  Salinity  pH  Chlorophy
```

Make hierfstat object

```
hf.filt <- genind2hierfstat(rad.filt, pop = c(rep("BIS", 10),rep("GB", 10),rep("NAR", 10), rep("PVD",10
```

```
hf.u <- genind2hierfstat(rad.u, pop = c(rep("BIS", 10),rep("GB", 10),rep("NAR", 10), rep("PVD",10)))
hf.u <- hf.u$hierfstat.no.imputation
```

Estimating effective migration surfaces (EEMS)

The program `runeems_snps` implements the EEMS method for analyzing spatial population structure. This version uses the pairwise genetic dissimilarity matrix computed from SNP data.

Here is the code used to run `runeems_snps` in both RStudio and command-line. Detailed steps for running this program can be accessed here. Input files are created in RStudio or manually (see below) and saved to the directory where `runeems_snps` will be run. The program is then run in the command-line. The outputs are then plotted in RStudio.

`runeems_snps` requires three data input files that have the same file name but different extension. The description below assumes that datapath is the full path + the file name (but without the extension).

1. datapath.diffs

`datapath.diffs` is the matrix of average pairwise genetic dissimilarities. This can be computed with bed2diffs from genetic data in plink binary format.

The dissimilarity matrix is nonnegative, symmetric, with 0s on the main diagonal. These conditions are necessary but not sufficient for diffs to be a valid dissimilarity matrix. Mathematically, diffs should be conditionally negative definite.

Steps for generating `datapath.diffs` are completed in RStudio.

```
# V1 methtod to get diffs matrix, preferred
bed2diffs_v1 <- function(Geno) {
  nIndiv <- nrow(Geno)
  nSites <- ncol(Geno)
  Diffs <- matrix(0, nIndiv, nIndiv)

  for (i in seq(nIndiv - 1)) {
    for (j in seq(i + 1, nIndiv)) {
      x <- Geno[i, ]
      y <- Geno[j, ]
      Diffs[i, j] <- mean((x - y)^2, na.rm = TRUE)
      Diffs[j, i] <- Diffs[i, j]
    }
  }
  Diffs
}
```

```
# V2 method to get .diffs matrix, only if V1 doesn't work
bed2diffs_v2 <- function(Geno) {
  nIndiv <- nrow(Geno)
  nSites <- ncol(Geno)
  Miss <- is.na(Geno)
  ## Impute NAs with the column means (= twice the allele frequencies)
  Mean <- matrix(colMeans(Geno, na.rm = TRUE), ## a row of means
                 nrow = nIndiv, ncol = nSites, byrow = TRUE) ## a matrix with nIndiv identical rows of r
  Mean[Miss == 0] <- 0 ## Set the means that correspond to observed genotypes to 0
  Geno[Miss == 1] <- 0 ## Set the missing genotypes to 0 (used to be NA)
  Geno <- Geno + Mean
  ## Compute similarities
  Sim <- Geno %*% t(Geno) / nSites
  SelfSim <- diag(Sim) ## self-similarities
  vector1s <- rep(1, nIndiv) ## vector of 1s
  ## This chunk generates a `diffs` matrix
  Diffs <- SelfSim %*% t(vector1s) + vector1s %*% t(SelfSim) - 2 * Sim
```

```
  Diffs
}
```

```
geno <- stratted.filt@tab
```

```
# Get rid of non-biallelic loci
multi.loci <- names(which(stratted.filt@loc.n.all != 2))
multi.cols <- which(grepl(paste0("^", multi.loci, "\\.\\d+$", collapse = "|"), colnames(geno)))
if (length(multi.cols)) geno <- geno[, - multi.cols]
nloci <- dim(geno)[2] / 2
dim(geno)
```

```
## [1]     40 181144
```

```
stopifnot(identical(stratted.filt@type, 'codom'))
```

```
# bed2diffs functions
diffs.v1 <- bed2diffs_v1(geno)
diffs.v2 <- bed2diffs_v2(geno)
# Round to 6 digits
diffs.v1 <- round(diffs.v1, digits = 6)
diffs.v2 <- round(diffs.v2, digits = 6)
```

Check that the dissimilarity matrix has one positive eigenvalue and nIndiv-1 negative eigenvalues, as required by a full-rank Euclidean distance matrix. If the V1 method does not make a Euclidean matrix, you must use V2.

```
tail(sort(round(eigen(diffs.v1)$values, digits = 2)))
```

```
## [1] -0.49 -0.49 -0.48 -0.48 -0.48 20.93
```

```
tail(sort(round(eigen(diffs.v2)$values, digits = 2)))
```

```
## [1] -0.48 -0.48 -0.47 -0.47 -0.46 20.62
```

```
# Set suffix for EEMS input files
suf <- "neutraldata-filt"
```

```
# This saves the file to directory
write.table(diffs.v1, paste(suf,".v1.diffs",sep=""),
            col.names = FALSE, row.names = FALSE, quote = FALSE)
```

2. datapath.coord

`datapath.coord` are the sample coordinates, two coordinates per sample, one sample per line. The sampling locations should be given in the same order as the rows and columns of the dissimilarity matrix.

Steps for generating `datapath.coord` are completed in RStudio.

```
## Get gps coordinates from previously created info matrix
xOR.info <- dplyr::filter(info)
gps_matrix <- select(xOR.info,c("Longitude","Latitude"))
```

```
#write .coord file
write.table(gps_matrix, paste(suf,".v1.coord",sep=""),col.names = FALSE, row.names = FALSE,quote = FALSE
```

3. datapath.outer

`datapath.outer` are the habitat coordinates, as a sequence of vertices that form a closed polygon. The habitat vertices should be listed counterclockwise and the first vertex should also be the last vertex, so

that the outline is a closed ring. Otherwise, EEMS attempts to "correct" the polygon and prints a warning message.

`datapath.outer` is created manually in Excel,based on site coordinates gathered from Google Maps, copied into terminal using `nano`, and saved as the file with the appropriate extension.

**`runeems_snps` is then run in command-line following the steps documented in NB_EEMS_OutlierHap.md

Back in RStudio to plot `runeems_snps` outputs

```
# Install rEEMSplots
library(rEEMSplots)
```

```
# Plotting EEMS after running runeems_snps
path = "./NB_EEMS_NeutralHap/"
dirs = c(paste0(path,"neutraldata-D200-chain1"), paste0(path,"neutraldata-D300-chain1"), paste0(path,"ne
```

```
eems.plots(mcmcpath = c(paste0(path,"./neutraldata-D200-chain1"), paste0(path,"neutraldata-D300-chain1"),
           longlat = T,add.grid=F,add.outline = T,add.demes = T,
           projection.in = "+proj=longlat +datum=WGS84",projection.out = "+proj=merc +datum=WGS84",
           add.map = T,add.abline = T, add.r.squared = T)
```

```
## Input projection: +proj=longlat +datum=WGS84
## Output projection: +proj=merc +datum=WGS84
```

```
## Loading rgdal (required by projection.in)
```

```
## Loading rworldmap (required by add.map)
```

```
## Loading rworldxtra (required by add.map)
```

```
## Using the default DarkOrange to Blue color scheme, with 'white' as the midpoint color.
## It combines two color schemes from the 'dichromat' package, which itself is based on
## a collection of color schemes for scientific data graphics:
##  Light A and Bartlein PJ (2004). The End of the Rainbow? Color Schemes for Improved Data
##  Graphics. EOS Transactions of the American Geophysical Union, 85(40), 385.
## See also http://geog.uoregon.edu/datagraphics/color_scales.htm
```

```
## Using 'euclidean' distance to assign interpolation points to Voronoi tiles.
```

```
## Processing the following EEMS output directories :
```

```
## ./NB_EEMS_NeutralHap/./neutraldata-D200-chain1./NB_EEMS_NeutralHap/neutraldata-D300-chain1./NB_EEMS_N
```

```
## Plotting effective migration surface (posterior mean of m rates)
```

```
## ./NB_EEMS_NeutralHap/./neutraldata-D200-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D300-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D600-chain1
```

```
## Using the default DarkOrange to Blue color scheme, with 'white' as the midpoint color.
## It combines two color schemes from the 'dichromat' package, which itself is based on
## a collection of color schemes for scientific data graphics:
##  Light A and Bartlein PJ (2004). The End of the Rainbow? Color Schemes for Improved Data
##  Graphics. EOS Transactions of the American Geophysical Union, 85(40), 385.
## See also http://geog.uoregon.edu/datagraphics/color_scales.htm
```

```
## Plotting effective diversity surface (posterior mean of q rates)
```

```
## ./NB_EEMS_NeutralHap/./neutraldata-D200-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D300-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D600-chain1
```

```
## Using the default DarkOrange to Blue color scheme, with 'white' as the midpoint color.
## It combines two color schemes from the 'dichromat' package, which itself is based on
## a collection of color schemes for scientific data graphics:
##  Light A and Bartlein PJ (2004). The End of the Rainbow? Color Schemes for Improved Data
##  Graphics. EOS Transactions of the American Geophysical Union, 85(40), 385.
## See also http://geog.uoregon.edu/datagraphics/color_scales.htm
```

```
## Plotting posterior probability trace
```

```
## ./NB_EEMS_NeutralHap/./neutraldata-D200-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D300-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D600-chain1
```

```
## Plotting average dissimilarities within and between demes
```

```
## ./NB_EEMS_NeutralHap/./neutraldata-D200-chain1
```

```
## ./NB_EEMS_NeutralHap/neutraldata-D300-chain1
```

```
## Using 'euclidean' distance to assign interpolation points to Voronoi tiles.
##
##
##
## Using 'euclidean' distance to assign interpolation points to Voronoi tiles.
```

```
## EEMS results for at least two different population grids
```

The plots are saved to the same directory where `runeems_snps` was run.

Pairwise Fst

```
fst.mat <- pairwise.WCfst(hf.filt)
```

```
gindF.fst.mat.triN <- as.matrix(fst.mat)
colnames(gindF.fst.mat.triN) <- pop_order
rownames(gindF.fst.mat.triN) <- pop_order
```

```
meltedN <- melt(gindF.fst.mat.triN, na.rm =TRUE)
round(gindF.fst.mat.triN,4)
```

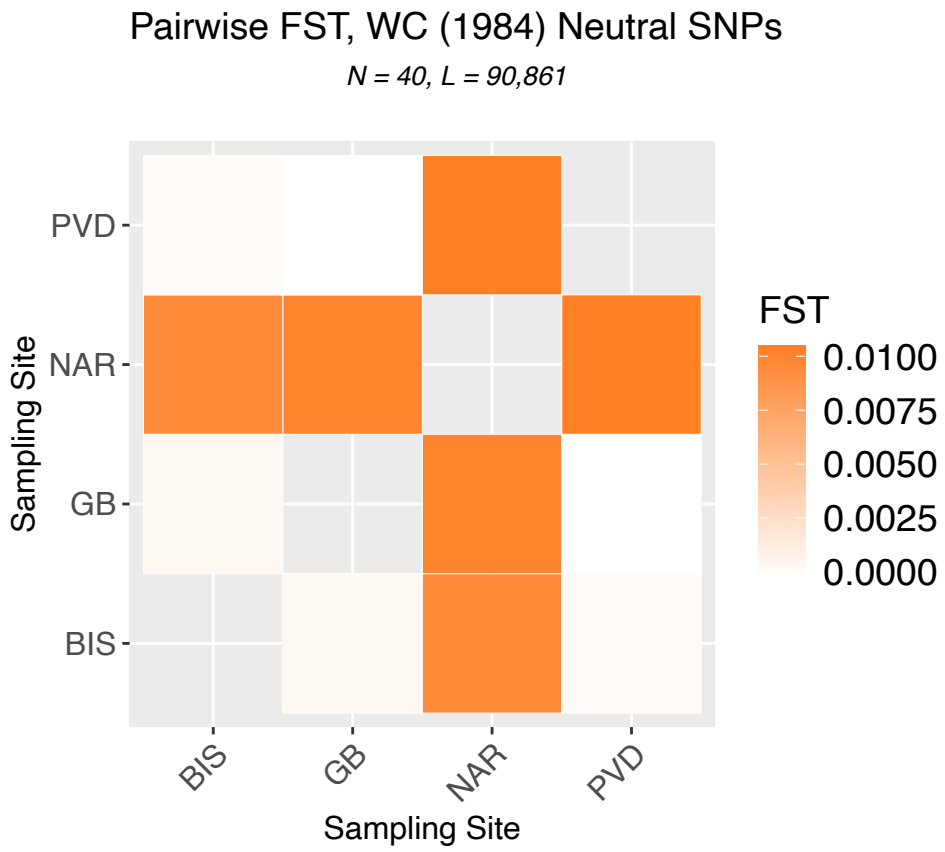
```
##        BIS      GB    NAR     PVD
## BIS     NA  0.0004 0.0095  0.0001
## GB  0.0004      NA 0.0101 -0.0002
## NAR 0.0095  0.0101     NA  0.0105
## PVD 0.0001 -0.0002 0.0105      NA
```

```
summary(meltedN$value)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0001853  0.0001348  0.0049696  0.0050672  0.0100511  0.0104632
```

```
#Plotting Pairwise fst
neutral <- ggplot(data = meltedN, aes(Var2, Var1, fill = value))+ geom_tile(color = "white")+
  scale_fill_gradient(low = "white", high = "chocolate1", name="FST")  +
  ggtitle(expression(atop("Pairwise FST, WC (1984) Neutral SNPs", atop(italic("N = 40, L = 90,861"), ""
  labs( x = "Sampling Site", y = "Sampling Site") +
```

```
  theme(axis.text.x = element_text(angle = 45, vjust = 1, size = 12, hjust = 1),axis.text.y = element_te
  theme(axis.title = element_text(size = 12),legend.text = element_text(size =14), legend.title = eleme
  theme(plot.title = element_text(size = 14)) +
coord_fixed()
neutral
```

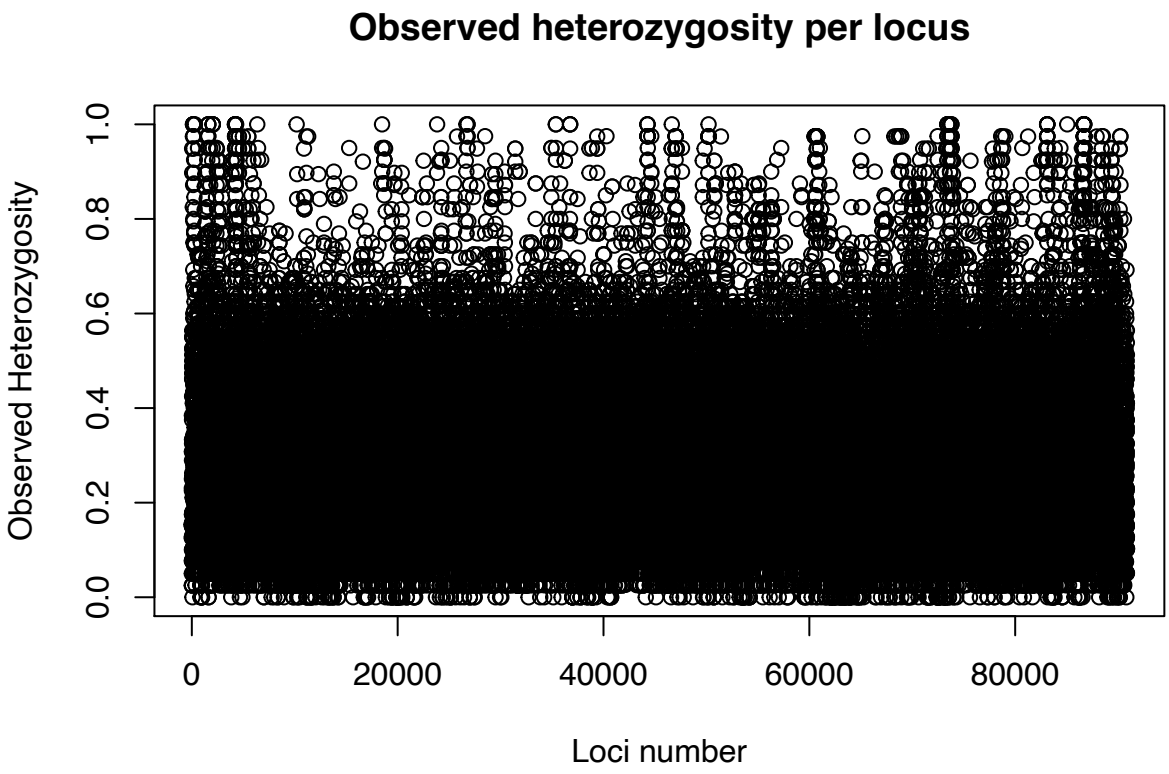


Genetic diversity (observed and expected heterozygosity)

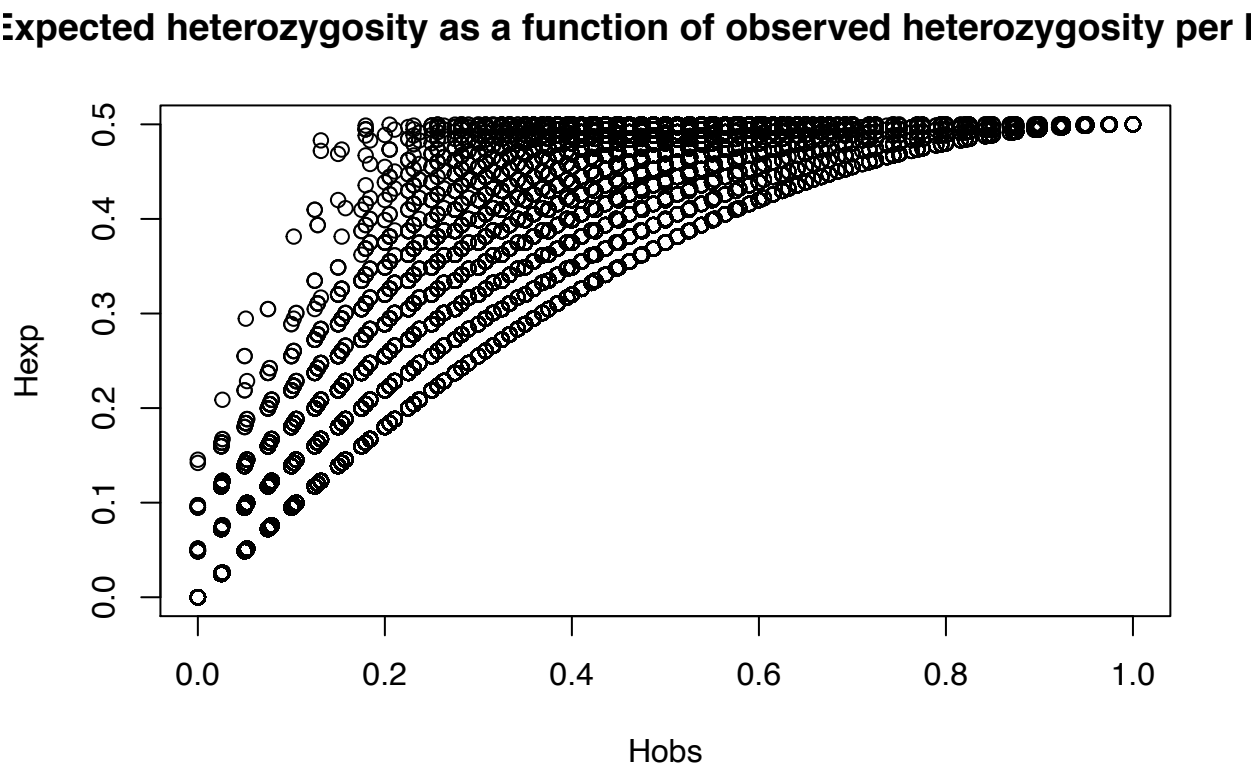
```
comb <- summary(stratted.filt)
names(comb)
```

```
## [1] "n"         "n.by.pop"  "loc.n.all" "pop.n.all" "NA.perc"   "Hobs"
## [7] "Hexp"
```

```
plot(comb$Hobs, xlab="Loci number", ylab="Observed Heterozygosity",
     main="Observed heterozygosity per locus")
```




```
plot(comb$Hobs,comb$Hexp, xlab="Hobs", ylab="Hexp",
     main="Expected heterozygosity as a function of observed heterozygosity per locus")
```




```
bartlett.test(list(comb$Hexp, comb$Hobs)) # a test : HO: Hexp = Hobs
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  list(comb$Hexp, comb$Hobs)
## Bartlett's K-squared = 2731.2, df = 1, p-value < 2.2e-16
```

Significant difference between Observed and expected heterzygosity.

```
basicstat <- basic.stats(hf.filt, diploid = TRUE, digits = 3)
```

```
as.data.frame(basicstat$overall)
```

```
##      basicstat$overall
## Ho               0.281
## Hs               0.273
## Ht               0.274
## Dst              0.001
## Htp              0.274
## Dstp             0.001
## Fst              0.004
## Fstp             0.005
## Fis             -0.028
## Dest             0.002
```

```
# get bootstrap confidence values for Fis
boot <- boot.ppfis(hf.filt,nboot = 1000)
```

```
boot5 <- boot.ppfis(hf.filt,nboot = 1000,quant = 0.5)
```

```
# add latitude for each population
latitude = c(41.545, 41.654, 41.505, 41.816)
```

```
# add longitude for each population
longitude = c(-71.431, -71.445, -71.453, -71.391)
```

```
# add distance for each population
distance = c(4.76, 0.47, 15.41, 1.49)
```

```
# add sewage effluent for each population
sewage = c(8.82, 14.60, 2.03, 59.86)
```

```
# add temperature for each population
temperature = c(23, 24, 25, 23)
```

```
# add salinity for each population
salinity = c(30, 28, 18, 25)
```

```
# add pH for each population
pH = c(7.9, 7.4, 7.6, 7.4)
```

```
# add Chlorophylla for each population
Chlor_a = c(4.9, 18.8, 4.6, 8.1)
```

```
# add DO for each population
DO = c(8.2, 5.7, 7, 4.9)
```

```
# combine all pop statistics
colnames(basicstat$Ho) <- pop_order
Ho <- colMeans(basicstat$Ho,na.rm = T)
He <- colMeans(basicstat$Hs,na.rm = T)
Fis<- boot5$fis.ci$ll
y <- cbind(pop_order,Ho,He,Fis,boot$fis.ci,latitude, longitude, distance,sewage, temperature, salinity,
y
```

```
##     pop_order        Ho        He     Fis      ll      hl latitude longitude
## BIS       BIS 0.2822666 0.2735835 -0.0318 -0.0347 -0.0293   41.545   -71.431
## GB         GB 0.2846475 0.2747413 -0.0361 -0.0387 -0.0335   41.654   -71.445
## NAR       NAR 0.2804854 0.2709379 -0.0353 -0.0380 -0.0326   41.505   -71.453
## PVD       PVD 0.2751708 0.2726034 -0.0096 -0.0121 -0.0070   41.816   -71.391
##     distance sewage temperature salinity  pH Chlor_a  DO
## BIS     4.76   8.82          23       30 7.9     4.9 8.2
## GB      0.47  14.60          24       28 7.4    18.8 5.7
## NAR    15.41   2.03          25       18 7.6     4.6 7.0
## PVD     1.49  59.86          23       25 7.4     8.1 4.9
```

```
summary(He)
```

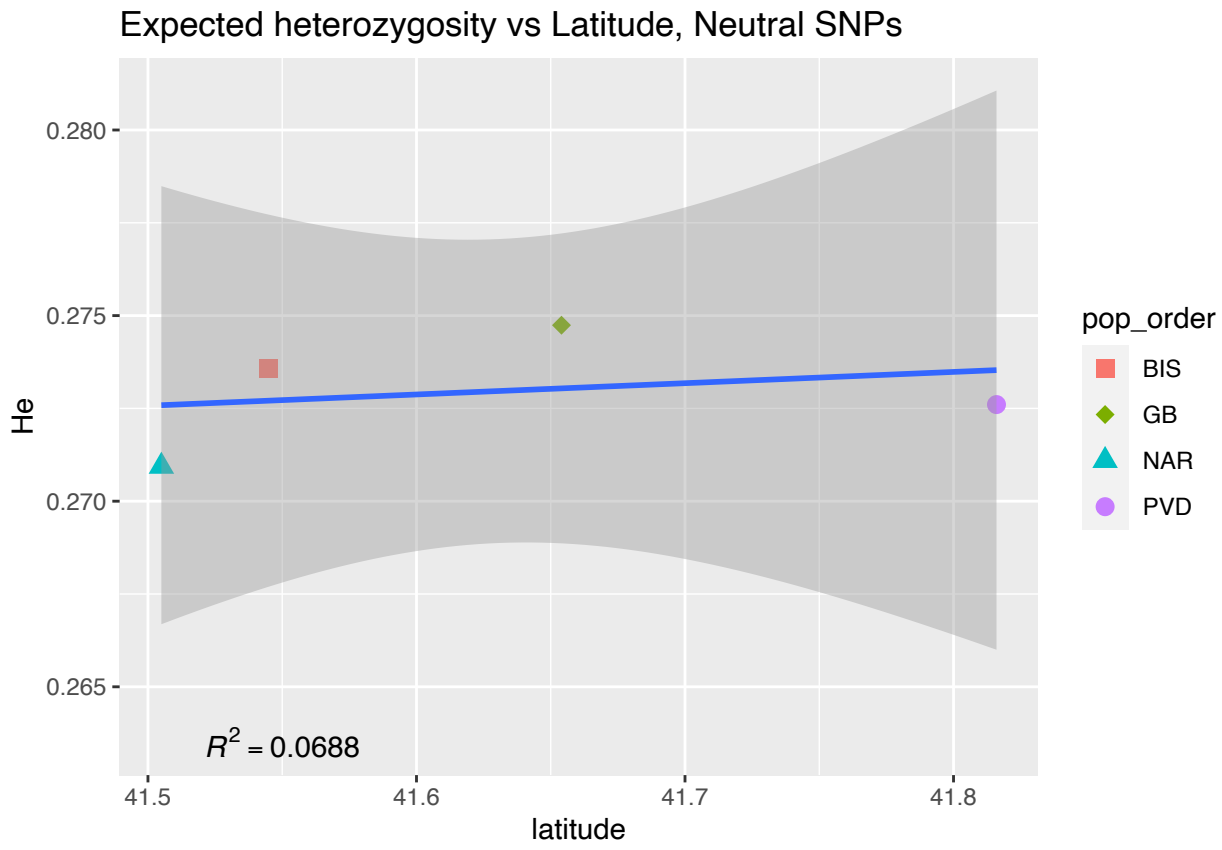
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2709  0.2722  0.2731  0.2730  0.2739  0.2747
```

```
summary(Fis)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.03610 -0.03550 -0.03355 -0.02820 -0.02625 -0.00960
```

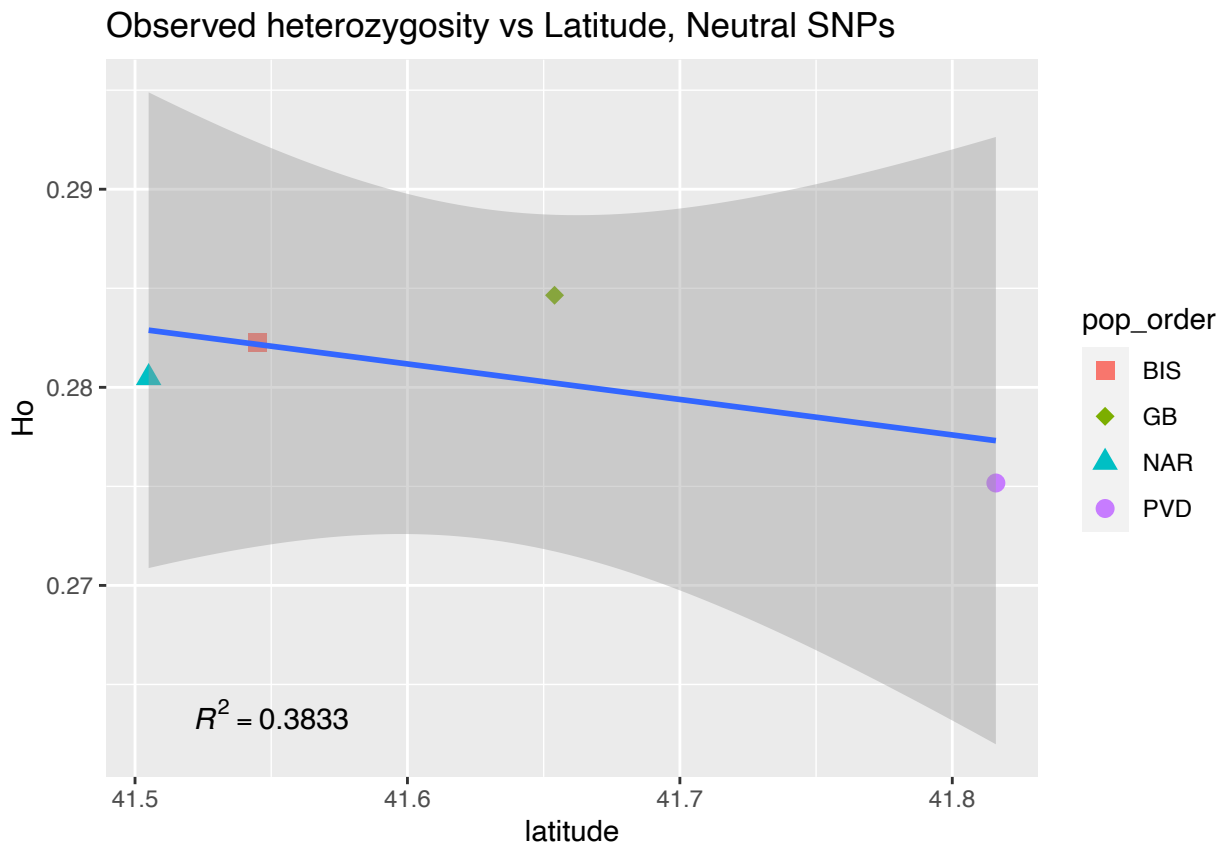
```
# Plot He vs Latitude
R2 = round(summary(lm(y$He ~ y$latitude))$r.squared, 4)
ggplot(y, aes(x = latitude, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Latitude, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R2), x=41.55, y=0.2635, parse=T) +
scale_x_continuous()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



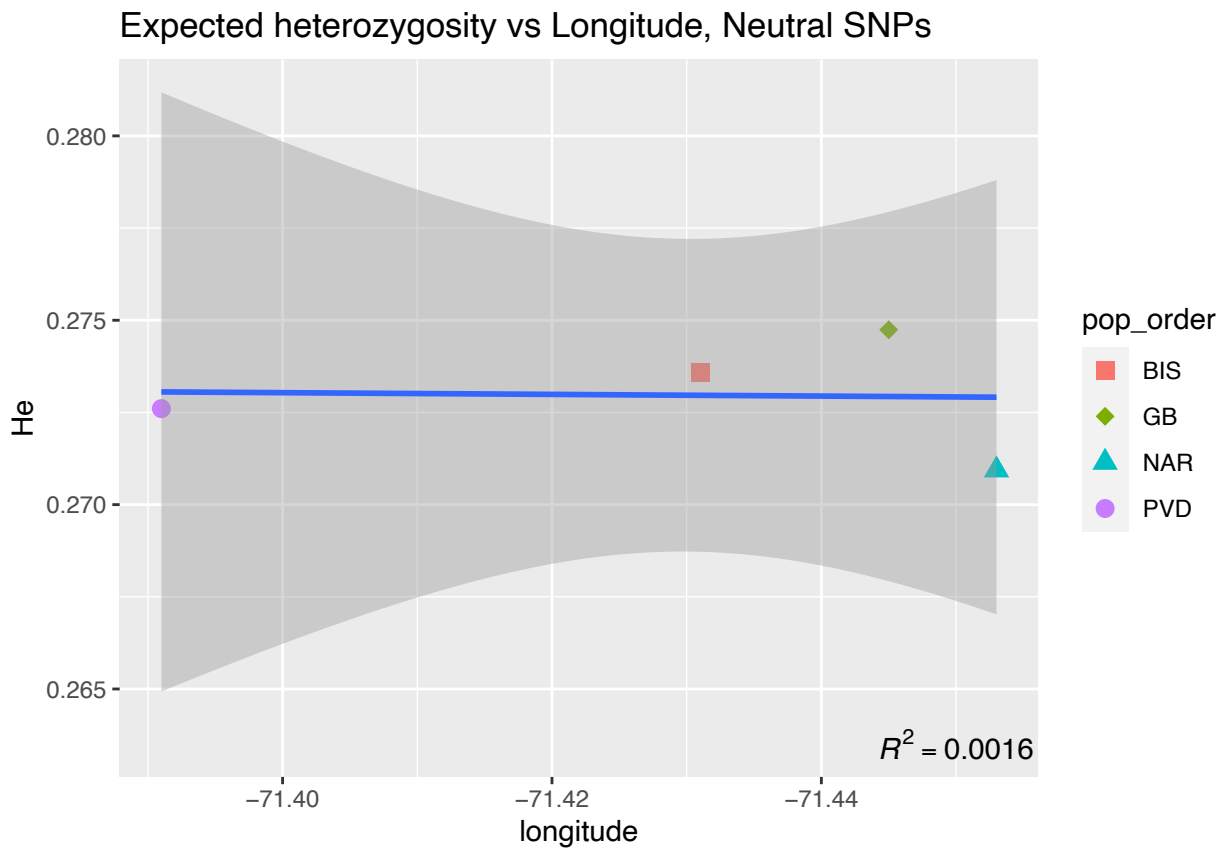

```
# Plot Ho vs Latitude
R2 = round(summary(lm(y$Ho ~ y$latitude))$r.squared, 4)
ggplot(y, aes(x = latitude, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Latitude, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R2), x=41.55, y=0.2635, parse=T) +
scale_x_continuous()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



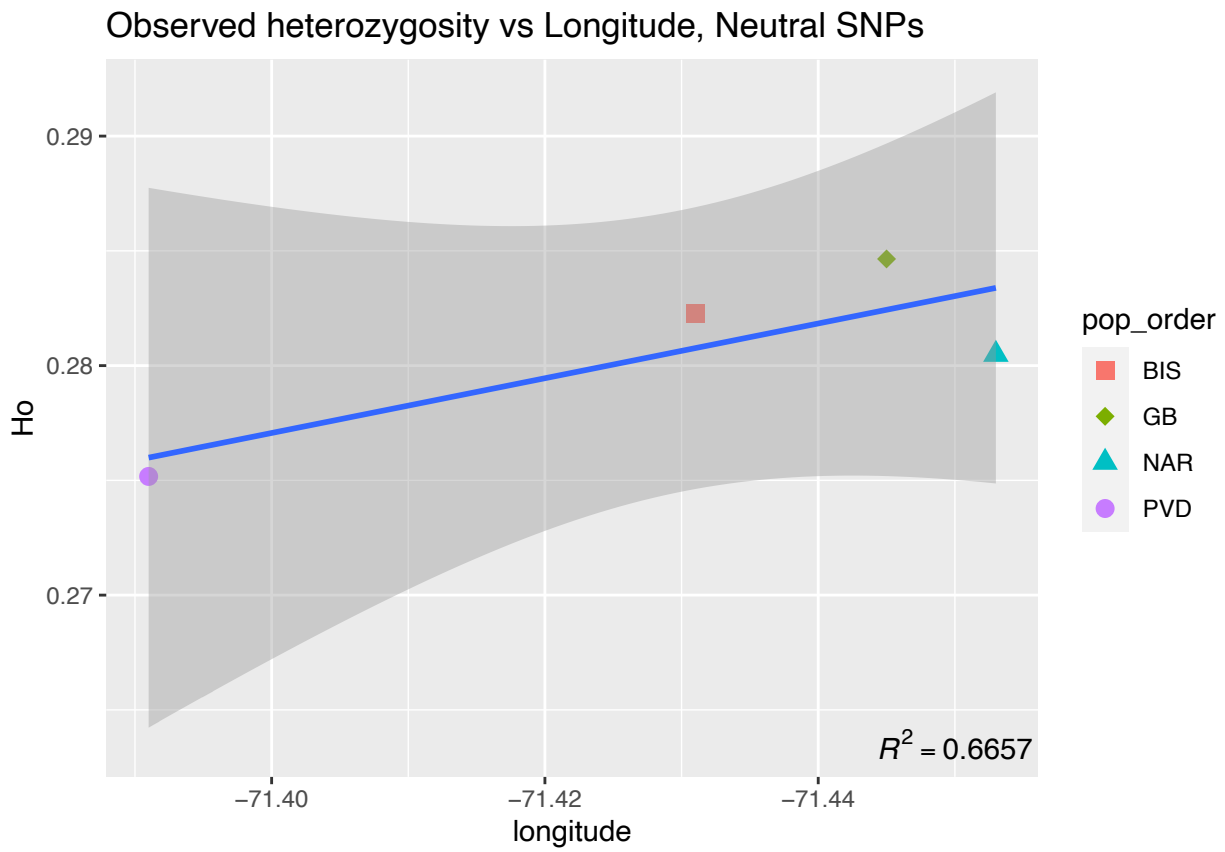

```
#Plot He vs Longitude
R3 = round(summary(lm(y$He ~ y$longitude))$r.squared, 4)
ggplot(y, aes(x = longitude, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Longitude, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R3), x=-71.45, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



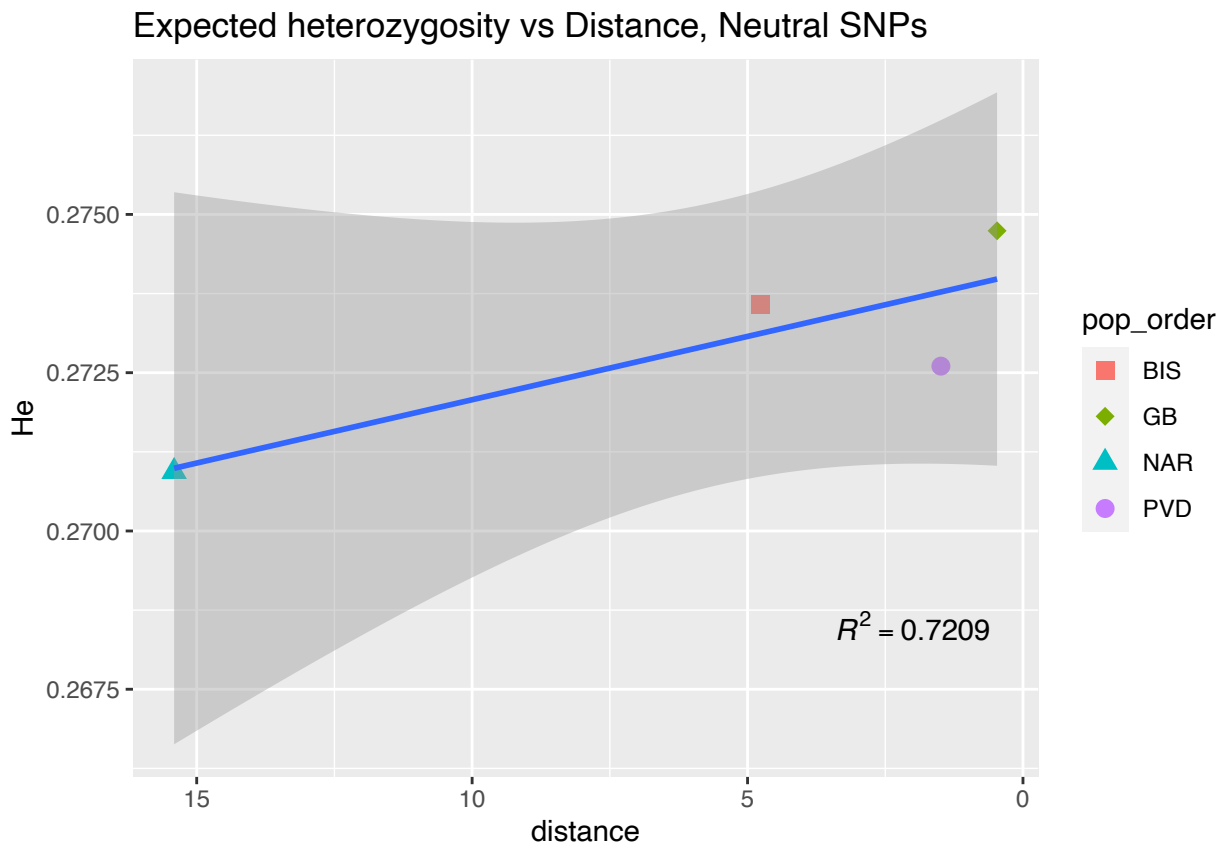

```
#Plot Ho vs Longitude
R3 = round(summary(lm(y$Ho ~ y$longitude))$r.squared, 4)
ggplot(y, aes(x = longitude, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Longitude, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R3), x=-71.45, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



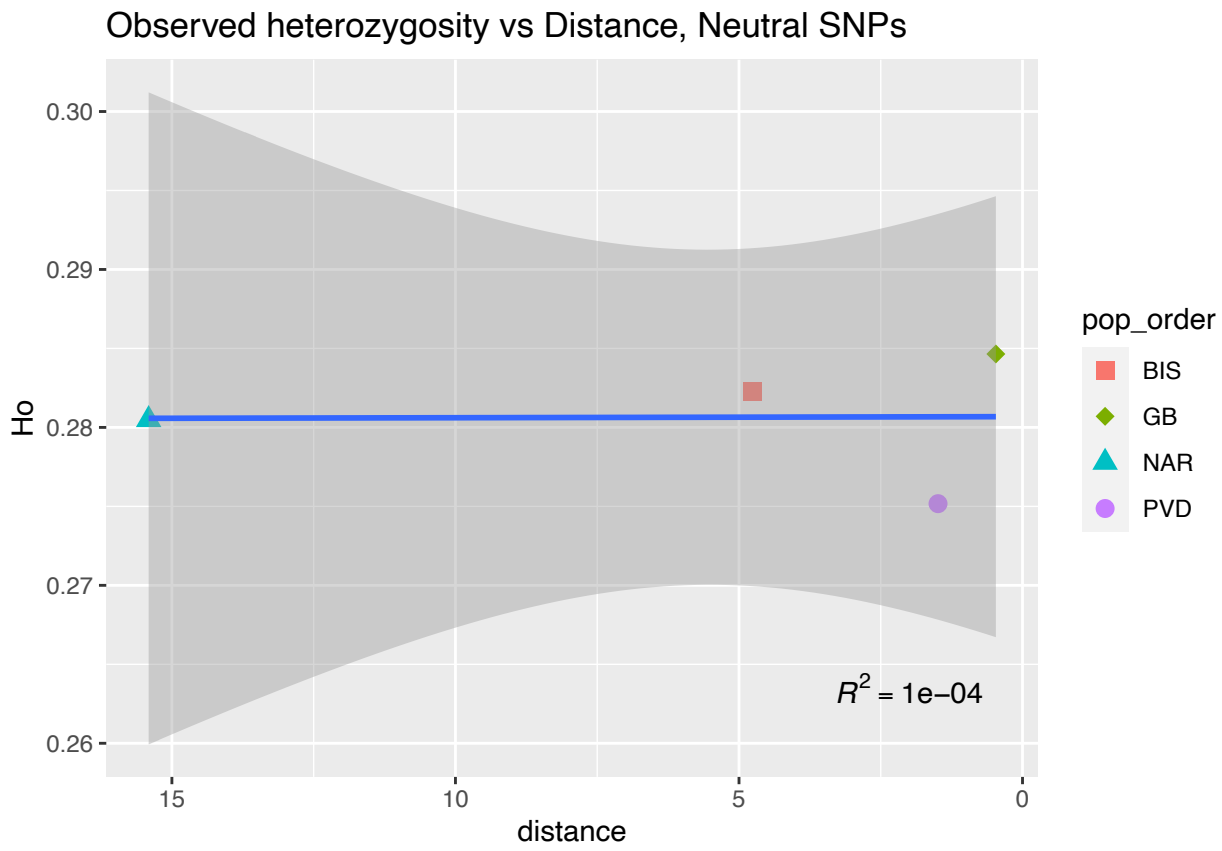

```
#Plot He vs Distance from sewage outflow
R4 = round(summary(lm(y$He ~ y$distance))$r.squared, 4)
ggplot(y, aes(x = distance, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Distance, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=2, y=0.2685, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



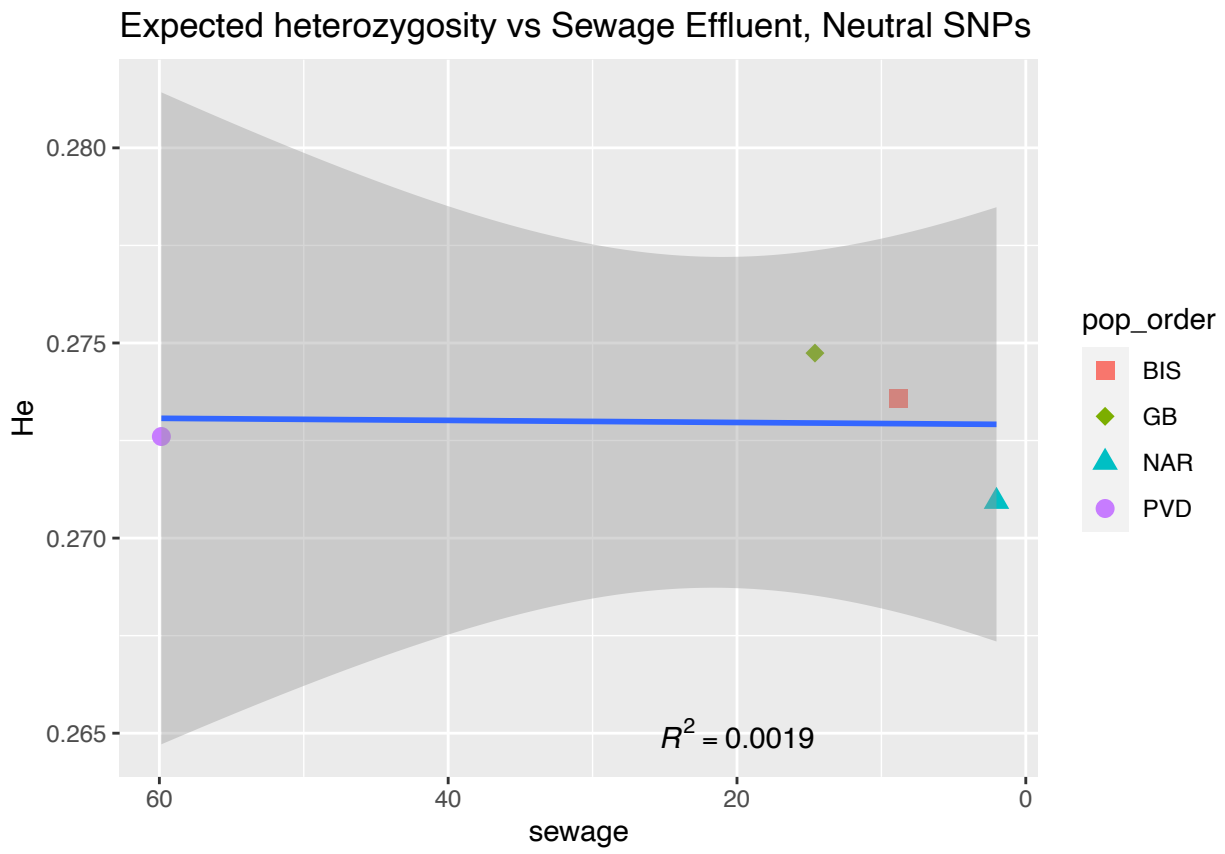

```
#Plot Ho vs Distance from sewage outflow
R4 = round(summary(lm(y$Ho ~ y$distance))$r.squared, 4)
ggplot(y, aes(x = distance, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Distance, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=2, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



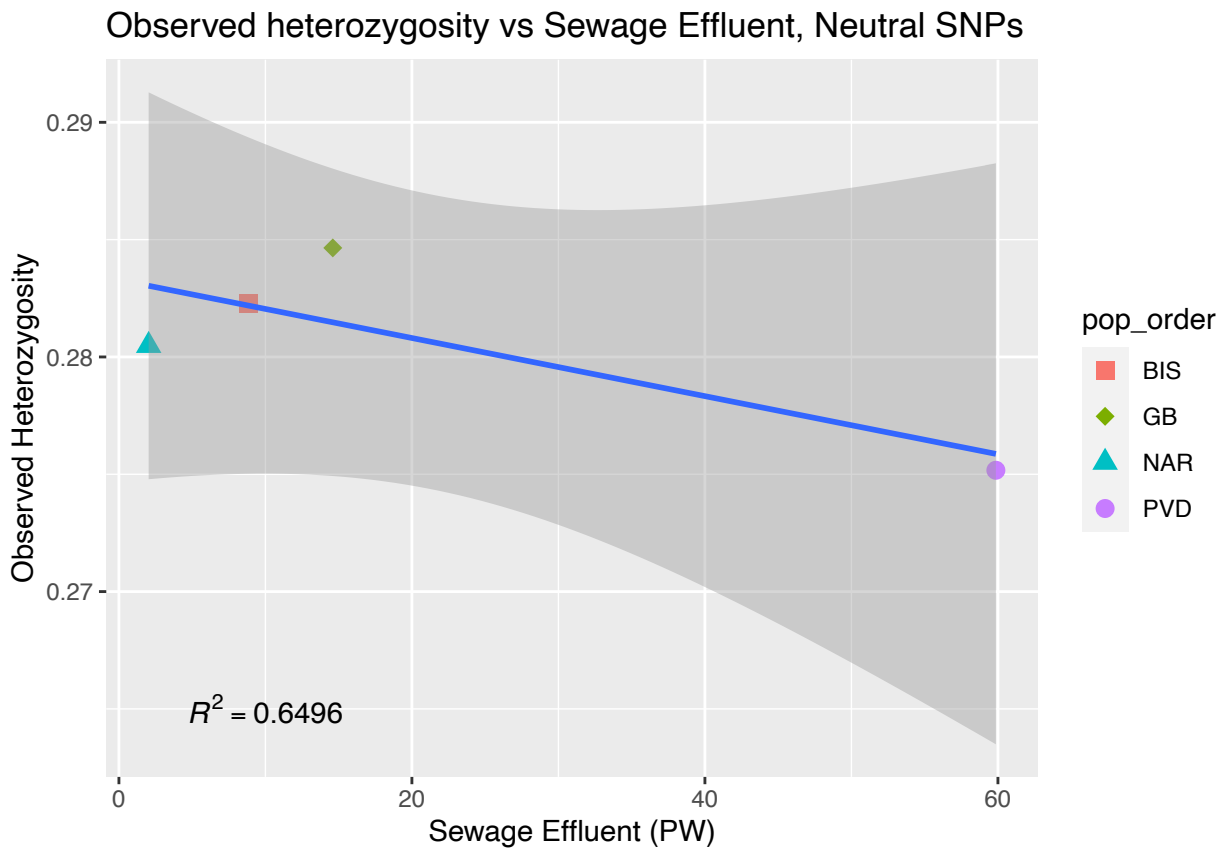

```
#Plot He vs Sewage Effluent
R4 = round(summary(lm(y$He ~ y$sewage))$r.squared, 4)
ggplot(y, aes(x = sewage, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Sewage Effluent, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=20, y=0.265, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



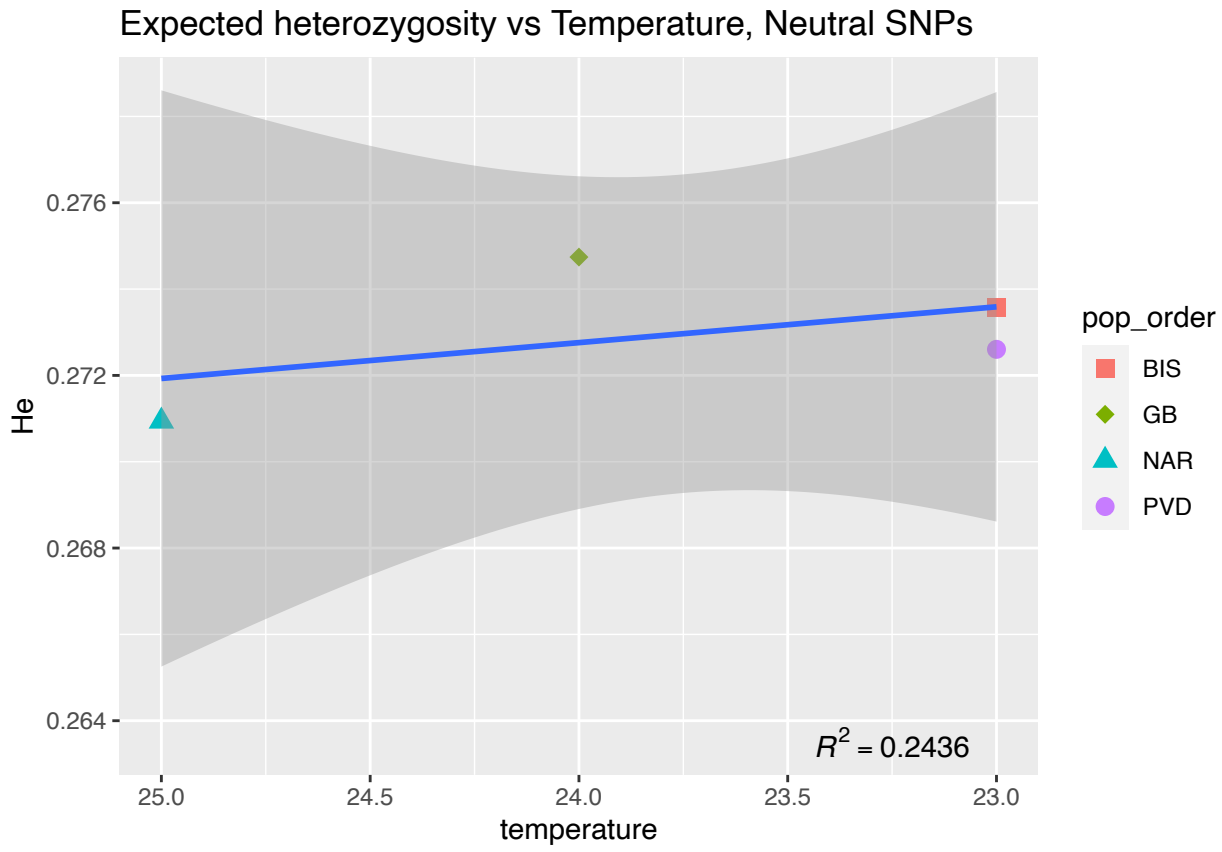

```
#Plot Ho vs Sewage Effluent
R4 = round(summary(lm(y$Ho ~ y$sewage))$r.squared, 4)
ggplot(y, aes(x = sewage, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Sewage Effluent, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=10, y=0.265, parse=T) +
scale_y_continuous("Observed Heterozygosity") + scale_x_continuous("Sewage Effluent (PW)")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



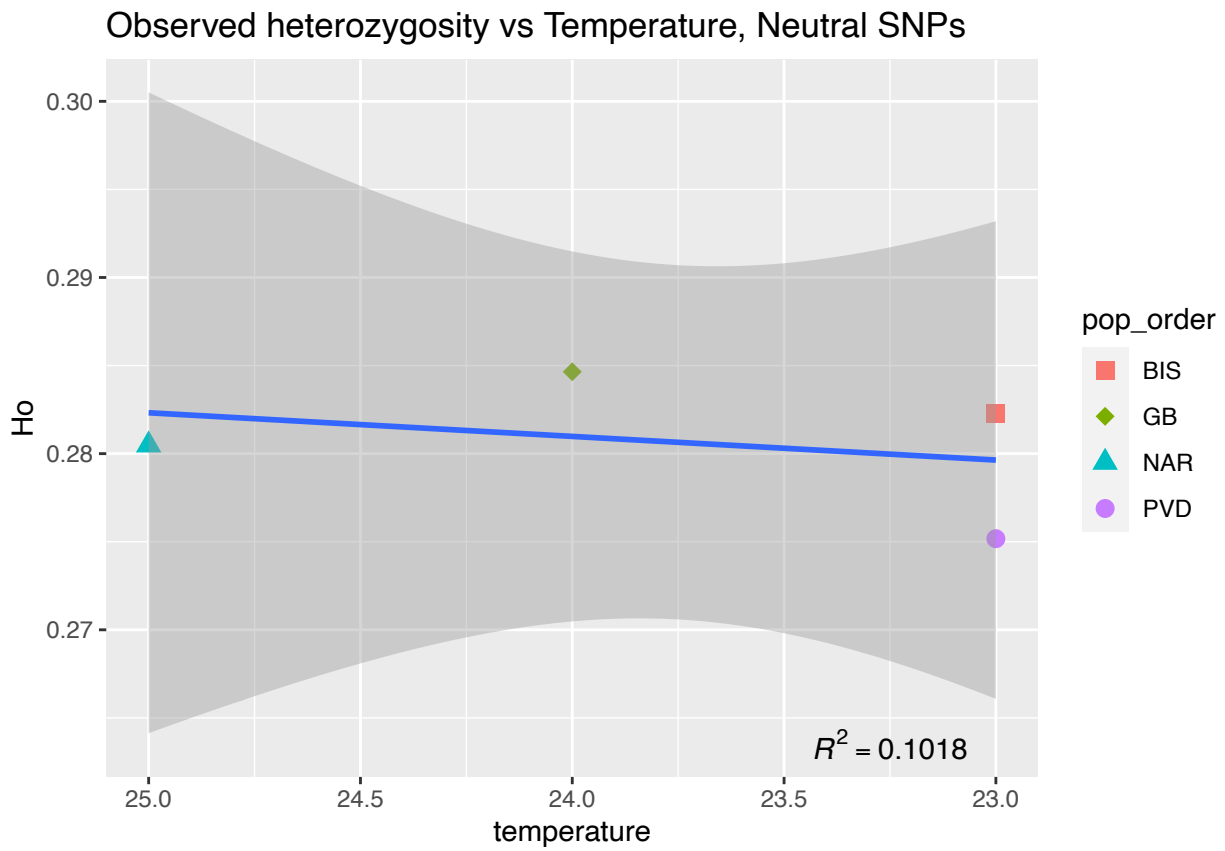

```
#Plot He vs Temperature
R4 = round(summary(lm(y$He ~ y$temperature))$r.squared, 4)
ggplot(y, aes(x = temperature, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Temperature, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=23.25, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



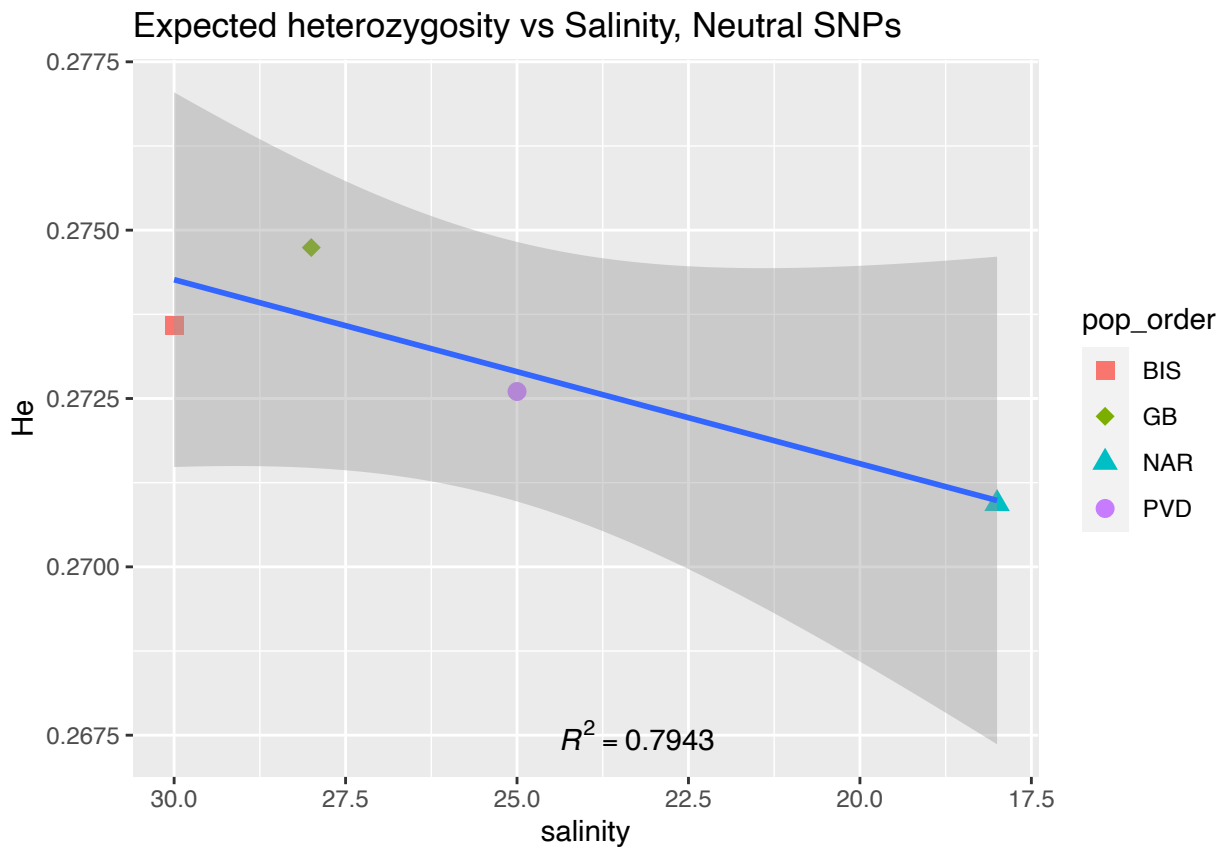

```
#Plot Ho vs Temperature
R4 = round(summary(lm(y$Ho ~ y$temperature))$r.squared, 4)
ggplot(y, aes(x = temperature, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Temperature, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=23.25, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



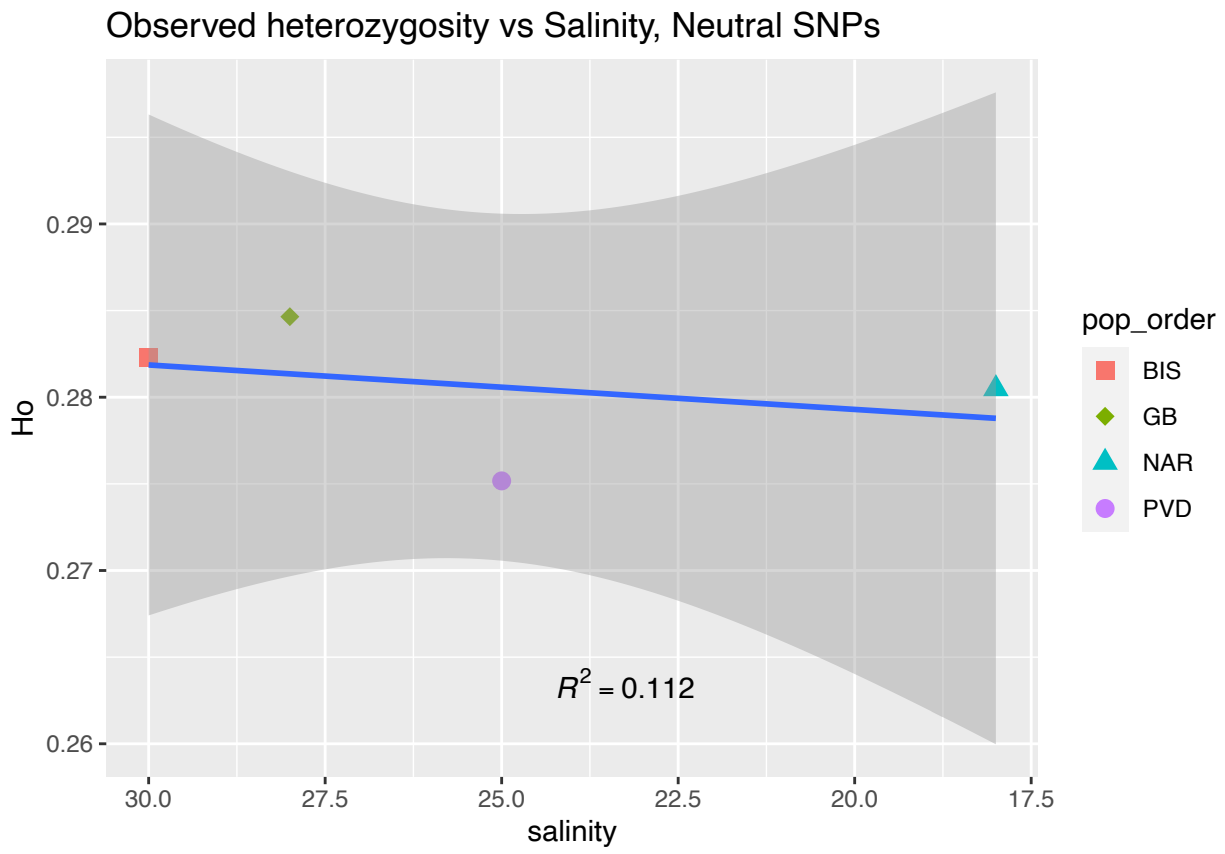

```
#Plot He vs Salinity
R4 = round(summary(lm(y$He ~ y$salinity))$r.squared, 4)
ggplot(y, aes(x = salinity, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Salinity, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=23.25, y=0.2675, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



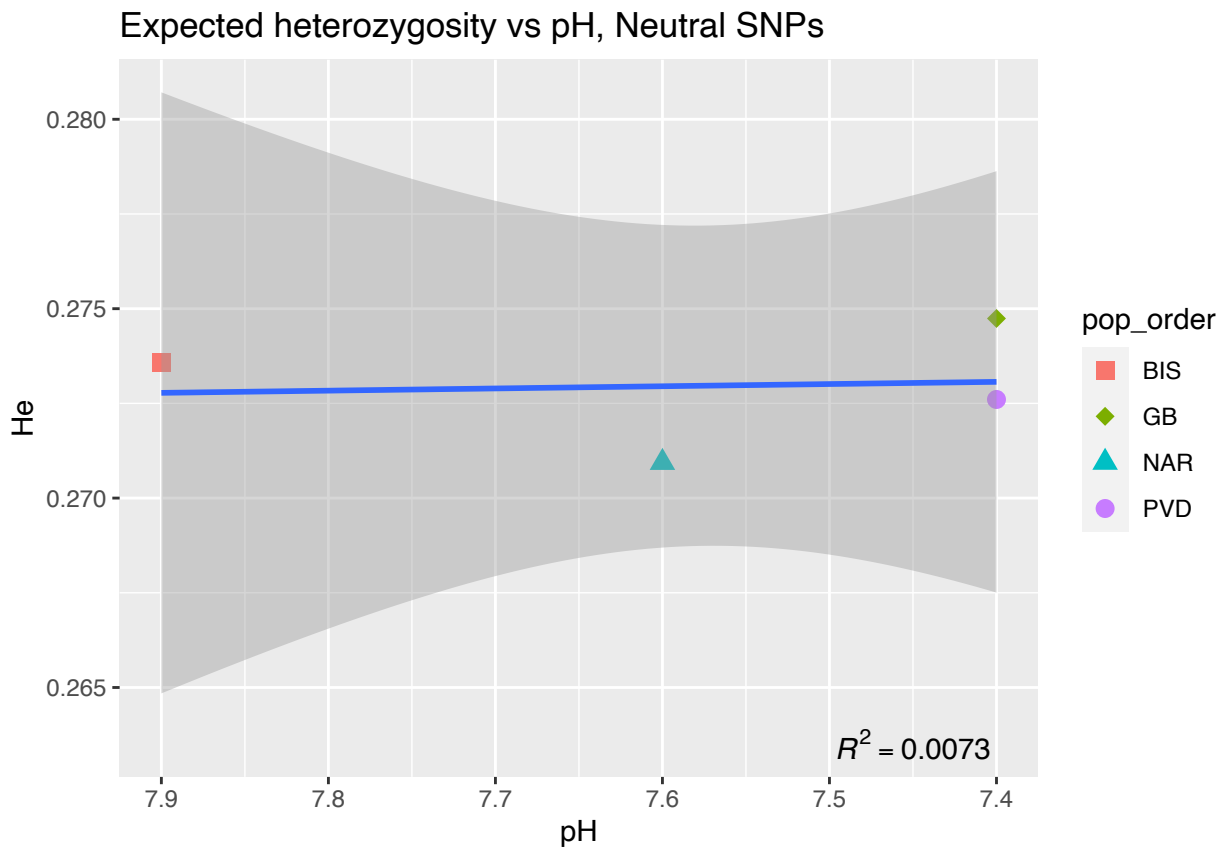

```
#Plot Ho vs Salinity
R4 = round(summary(lm(y$Ho ~ y$salinity))$r.squared, 4)
ggplot(y, aes(x = salinity, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Salinity, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=23.25, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



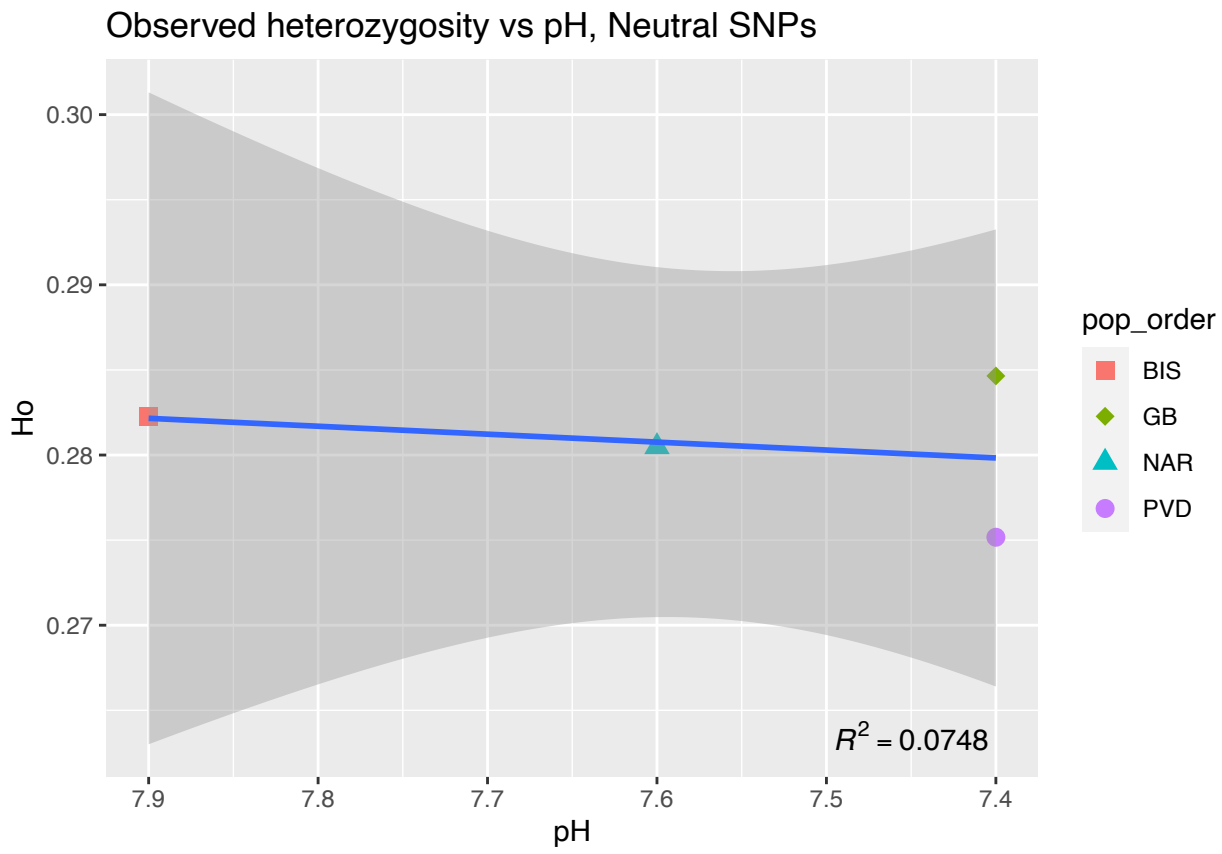

```
#Plot He vs pH
R4 = round(summary(lm(y$He ~ y$pH))$r.squared, 4)
ggplot(y, aes(x = pH, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs pH, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=7.45, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



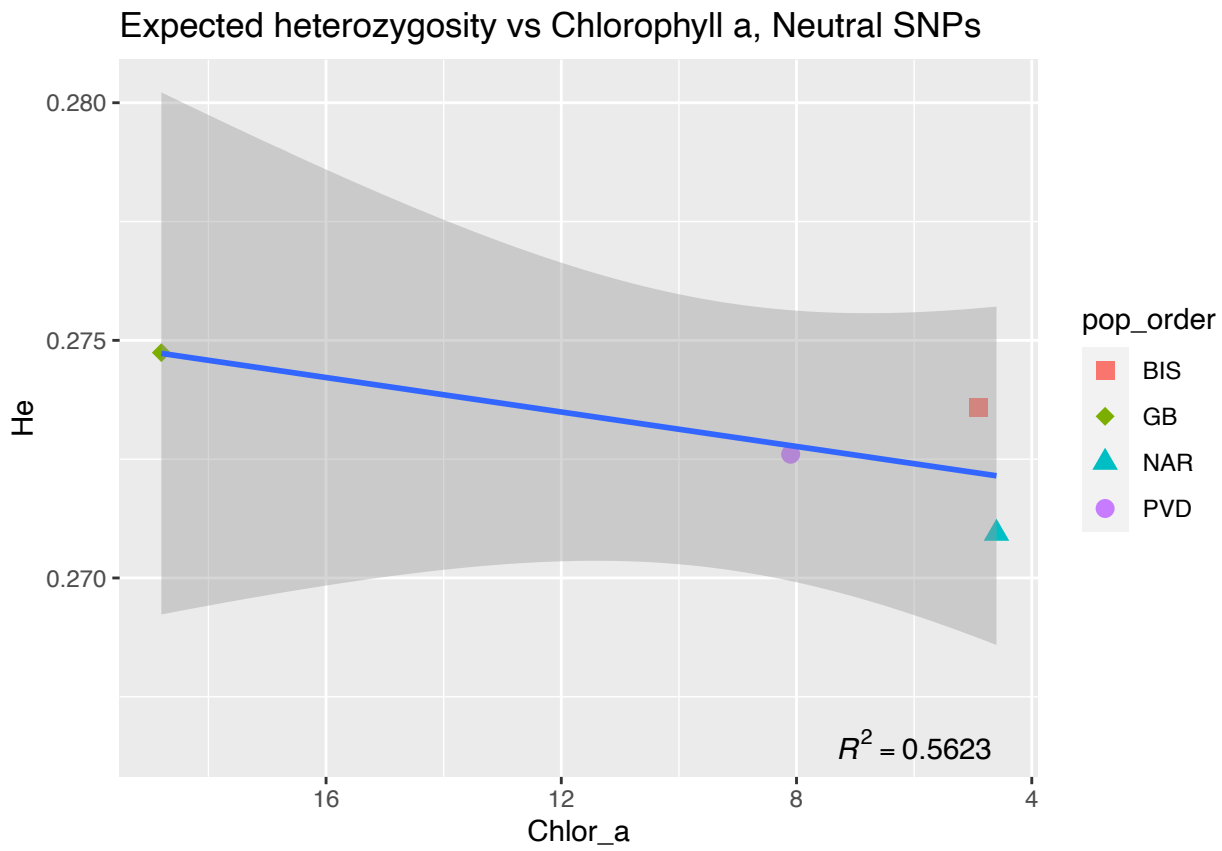

```
#Plot Ho vs pH
R4 = round(summary(lm(y$Ho ~ y$pH))$r.squared, 4)
ggplot(y, aes(x = pH, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs pH, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=7.45, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



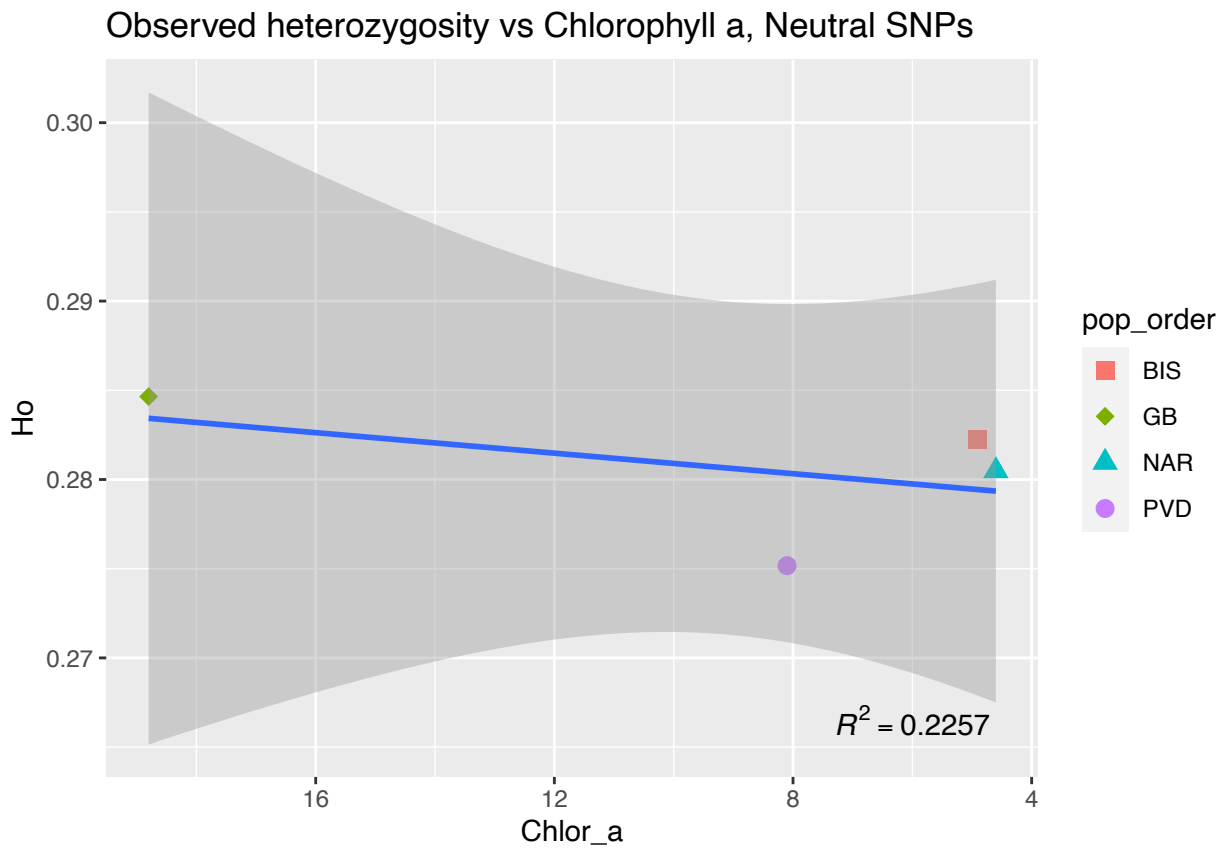

```
#Plot He vs Chlorophyll a
R4 = round(summary(lm(y$He ~ y$Chlor_a))$r.squared, 4)
ggplot(y, aes(x = Chlor_a, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Chlorophyll a, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=6, y=0.2665, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



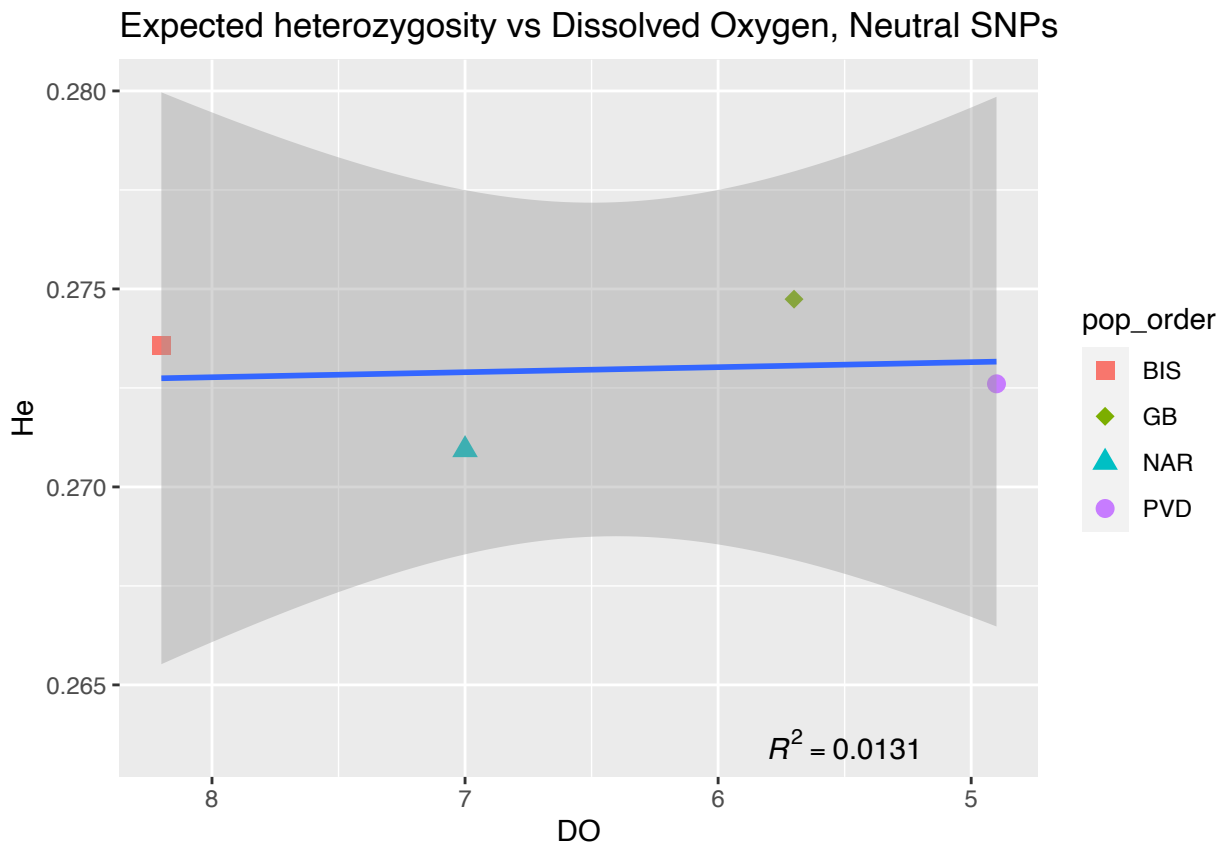

```
#Plot Ho vs Chlorophyll a
R4 = round(summary(lm(y$Ho ~ y$Chlor_a))$r.squared, 4)
ggplot(y, aes(x = Chlor_a, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Chlorophyll a, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=6, y=0.2665, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



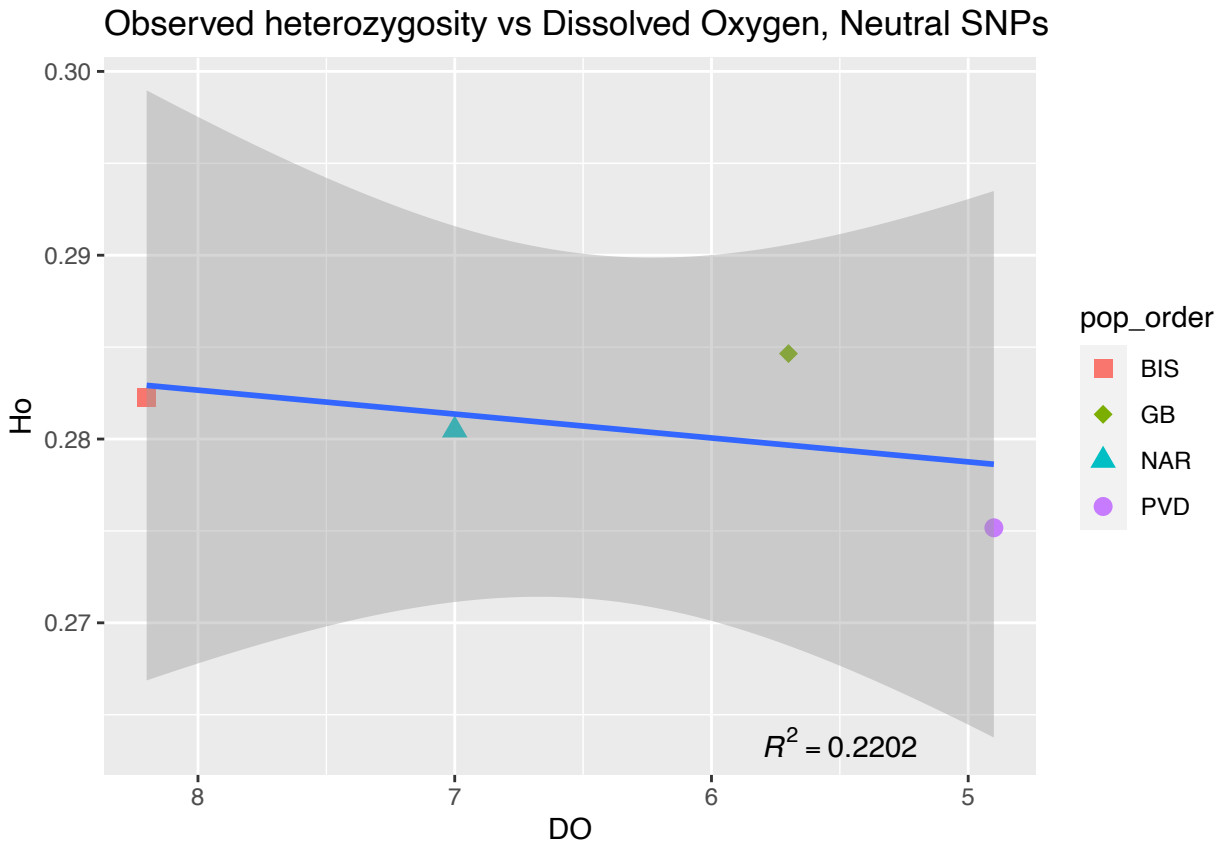

```
#Plot He vs Dissolved Oxygen
R4 = round(summary(lm(y$He ~ y$DO))$r.squared, 4)
ggplot(y, aes(x = DO, y = He)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Expected heterozygosity vs Dissolved Oxygen, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=5.5, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



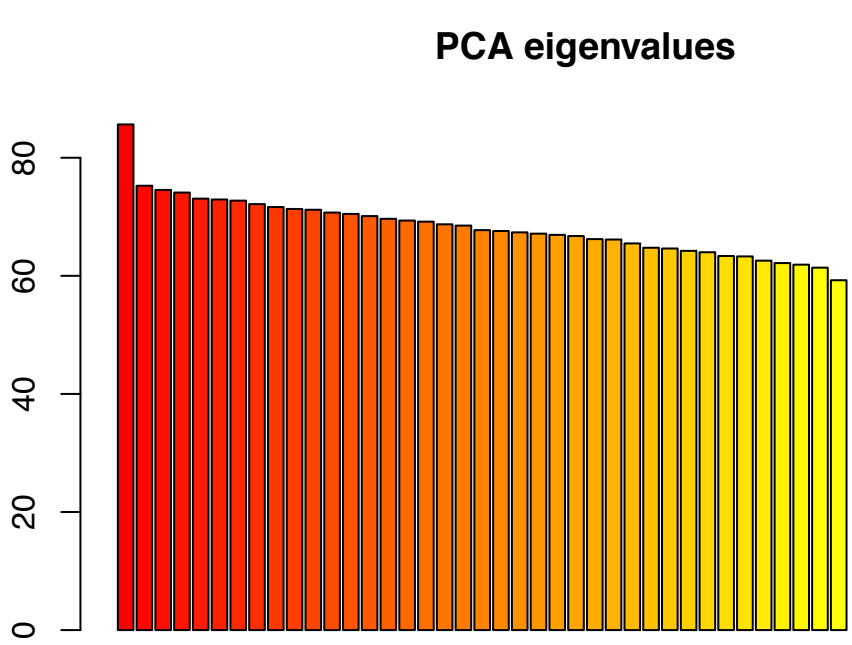

```
#Plot Ho vs Dissolved Oxygen
R4 = round(summary(lm(y$Ho ~ y$DO))$r.squared, 4)
ggplot(y, aes(x = DO, y = Ho)) + geom_point(aes(shape=pop_order, color=pop_order), size = 3) +
  scale_shape_manual(values = c(15,18,17,16)) +
geom_smooth(method=lm) +
ggtitle("Observed heterozygosity vs Dissolved Oxygen, Neutral SNPs") +
annotate(geom = "text", label=paste("italic(R^2)==",R4), x=5.5, y=0.2635, parse=T) +
scale_x_reverse()
```

```
## `geom_smooth()` using formula 'y ~ x'
```

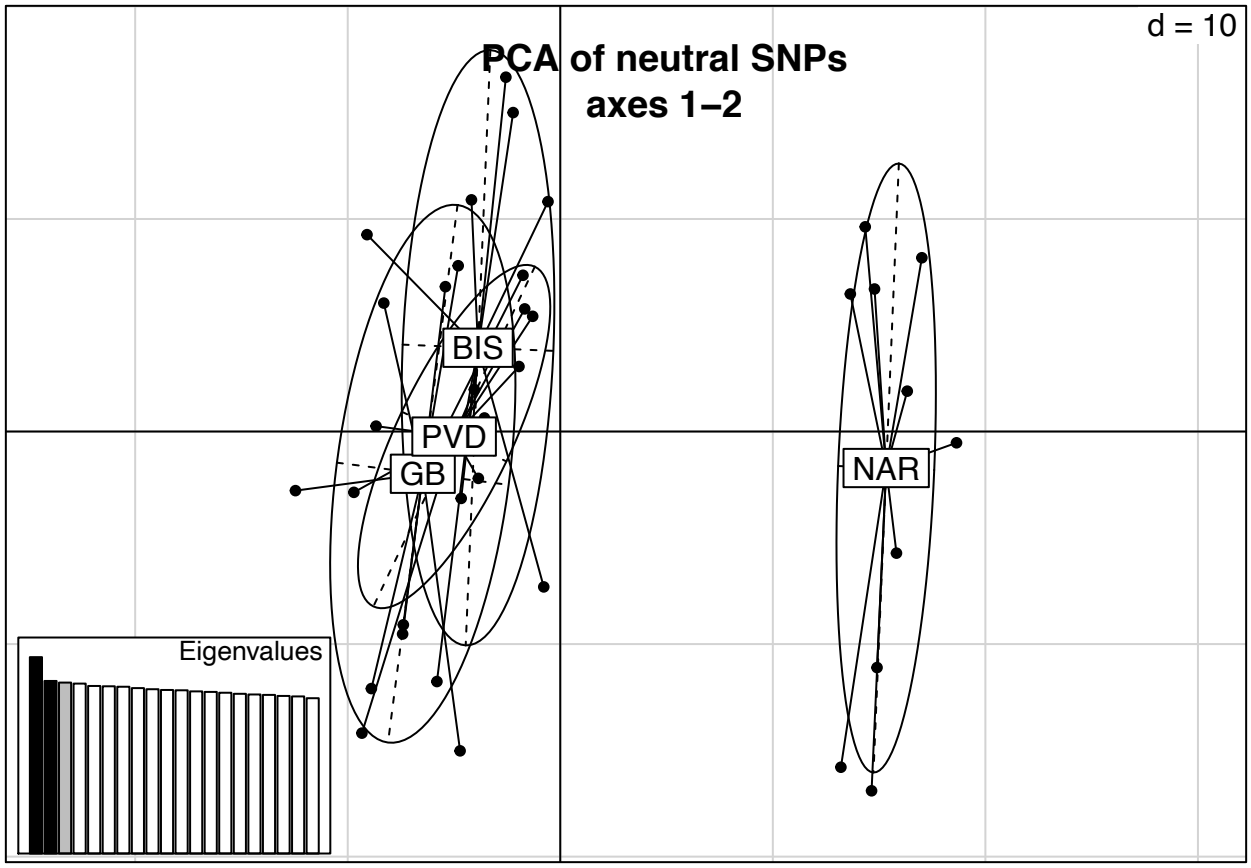


PCA

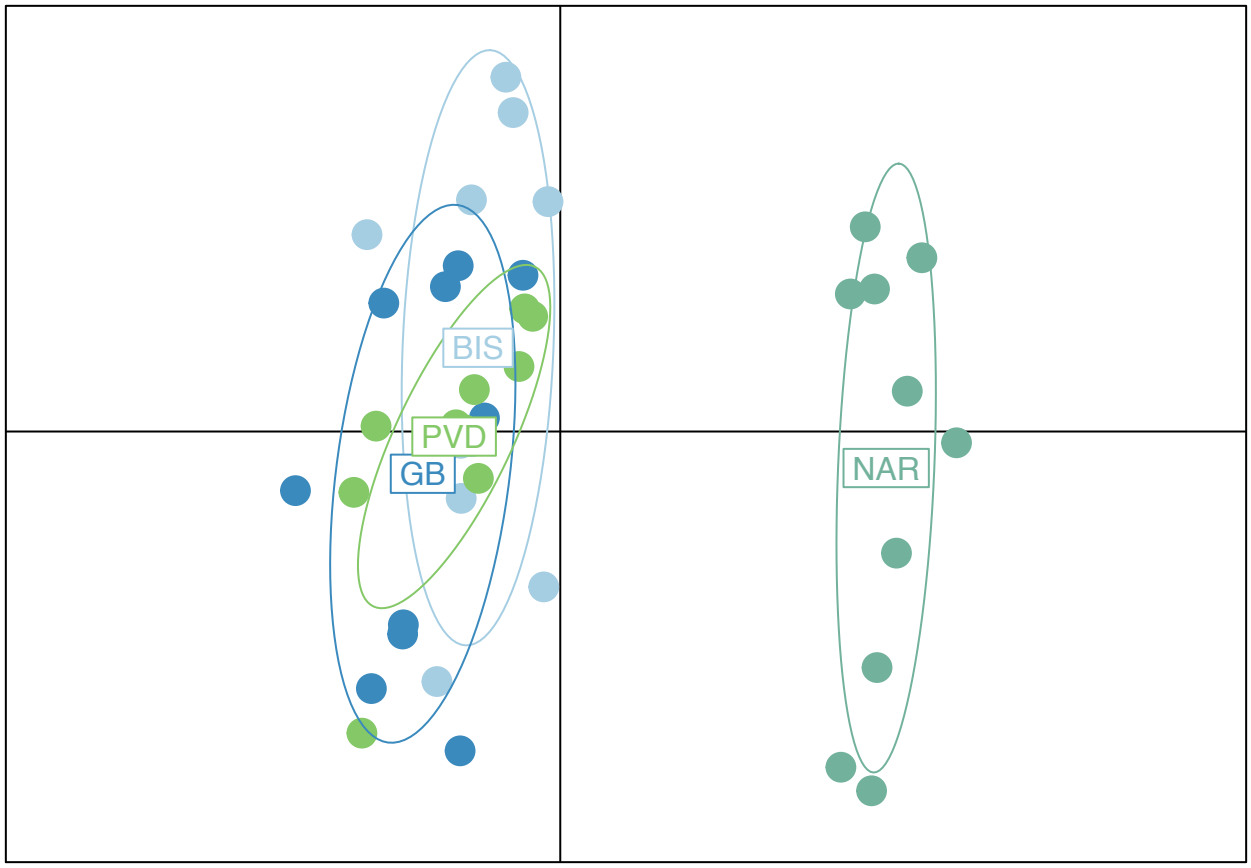
```
X <- tab(stratted.u, freq = TRUE, NA.method = "mean")
pca1 <- dudi.pca(X, scale = FALSE, scannf = FALSE, nf = 3)
barplot(pca1$eig[1:50], main = "PCA eigenvalues", col = heat.colors(50))
```




```
s.class(pca1$li, pop(stratted.filt))
title("PCA of neutral SNPs\naxes 1-2")
add.scatter.eig(pca1$eig[1:20], 3,1,2)
```




```
col <- funky(15)
s.class(pca1$li, pop(stratted.filt),xax=1,yax=2, col=col, axesell=FALSE, cstar=0, cpoint=3, grid=FALSE)
```



Seascape Redundancy Analysis

This code follows that documented by Tom Jenkins.

Prepare genetic data for redundancy analysis.

Notes before execution:

1. Make sure all required R packages are installed.
2. Set working directory to the location of this R script.

```
# Load packages
library(adegenet)
library(poppr)
```

```
## Registered S3 method overwritten by 'pegas':
##   method      from
##   print.amova ade4
```

```
## This is poppr version 2.8.5. To get started, type package?poppr
## OMP parallel support: available
```

```
##
## Attaching package: 'poppr'
```

```
## The following object is masked from 'package:radiator':
```

```
##
##     private_alleles
```

```
library(dplyr)
library(reshape2)
library(ggplot2)
library(vcfR)
```

```
# Explore data
rad.filt
```

```
## /// GENIND OBJECT /////////
##
##  // 40 individuals; 90,861 loci; 181,433 alleles; size: 82.8 Mb
##
##  // Basic content
##    @tab:  40 x 181433 matrix of allele counts
##    @loc.n.all: number of alleles per locus (range: 1-2)
##    @loc.fac: locus factor for the 181433 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: adegenet::df2genind(X = t(x), sep = sep, pop = ..2, strata = ..1)
##
##  // Optional content
##    @pop: population of each individual (group size range: 10-10)
##    @strata: a data frame with 12 columns ( Individual, Population, Latitude, Longitude, Distance, SE
```

```
nLoc(rad.filt) # number of loci
```

```
## [1] 90861
```

```
nPop(rad.filt) # number of sites
```

```
## [1] 4
```

```
nInd(rad.filt) # number of individuals
```

```
## [1] 40
```

```
summary(rad.filt$pop) # sample size
```

```
## BIS  GB NAR PVD
##  10  10  10  10
```

```
# Calculate allele frequencies for each site
allele_freqs = data.frame(rraf(rad.filt, by_pop = TRUE, correction = FALSE), check.names = FALSE)
```

```
# Keep only the first of the two alleles for each SNP (since p=1-q).
allele_freqs = allele_freqs[, seq(1, dim(allele_freqs)[2], 2)]
```

```
# Export allele frequencies
write.csv(allele_freqs, file = "all_allele_freqs.csv", row.names = TRUE)
```

Calculate minor allele frequencies

```
# Separate genind object by site
site_list = seppop(rad.filt)
names(site_list)
```

```
## [1] "BIS" "GB"  "NAR" "PVD"
```

```
# Calculate the minor allele frequency for each site
maf_list = lapply(site_list, FUN = minorAllele)
```

```
# Convert list to dataframe
maf = as.data.frame(maf_list) %>% t() %>% as.data.frame()
```

```
# Export minor allele frequencies
write.csv(maf, file = "minor_allele_freqs.csv", row.names = TRUE)
```

Prepare environmental data for redundancy analysis.

Environmental variables:

- Distance from sewage effluent source (km)
- Sewage Effluent (PW stats)
- Mean temperature (deg C)
- Mean Salinity (psu)
- Mean pH
- Mean Chlorophyll-a (ug/L)
- Mean Dissolved Oxygen (mg/L)

Environmental data for each population is saved in a `strata_pop` file that can be accessed here

```
# All environmental data was previously saved in strata file
strata_pop <- read.table("strata_pop", header=TRUE)
strata_pop
```

```
##   Population Latitude Longitude    Sewage Temperature Salinity  pH Chlorophylla
## 1        BIS   41.545   -71.431  8.824636          23       30 7.9          4.9
## 2         GB   41.654   -71.445 14.596049          24       28 7.4         18.8
## 3        NAR   41.505   -71.453  2.027484          25       18 7.6          4.6
## 4        PVD   41.816   -71.391 59.860038          23       25 7.4          8.1
##    DO
## 1 8.2
## 2 5.7
## 3 7.0
```

```
## 4 4.9
```

```
# Export data as a csv file
write.csv(strata_pop, file="environmental_data.csv", row.names = FALSE)
```

I also prepared spatial data for the redundancy analysis which is documented here.

Allele frequency, environmental, and spatial csv files are saved to your working directory and must be imported into the Rscript to run the redundancy analysis. Documentation of the RDA can be accessed here.

RDA and pRDA output plots can be viewed here.