Scripts

Run code one chunk at the time, The server may return errors otherwise

1. PCA

Load data, define functions needed

```
library(snpMatrix)

#### 1. Load data

## Retrieve information about species from popinfo file
popinfo = read.table("grants_popinfo.txt")
# To see the species represented in the data and how many individuals are there
table(popinfo[,2])
# Select species column
species = popinfo[2:nrow(popinfo),2]



## Load genotype data (use all grants file bim, bed and fam)
data <- read.plink("grants")
geno <- matrix(as.integer(data@.Data), nrow = nrow(data@.Data))
dim(geno) # 95 rows (individuals),35174 columns (SNPs)


#### 2. Functions needed for PCA

## 2.1 Preprocessing

# Sort the individuals by species and remove outgroup if needed
preprocess <- function(geno, remove_outgroup = FALSE){
  # Add species as last column
  geno <- cbind(geno, species)
  # Sort them by species
  geno <- geno[order(species),]  # 2 = 37 x g.granti
                                 # 3 = 26 x robertse
                                 # 4 = 10 x notata
                                 # 5 = 19 x petersii
                                 # 6 = 3  x thomson
  if (remove_outgroup == TRUE){
    # Remove the outgroup
    geno <- geno[geno[,ncol(geno)] != 6,]
  }
  # Remove species column
  geno <- geno[,-ncol(geno)]
return(geno)
```

```
}

# Change values 0 to NA and change genotypes to 0, 1, 2
detect_NA <- function(geno){
  geno <- t(geno)                # transpose the matrix -> 35174 rows (SNPs),
                                 #                         95 columns (individuals)
  geno[geno == 0] <- NA          # genotype 0 = NA values
  geno <- geno - 1               # now the genotypes are 0, 1, 2
  return(geno)
}

## 2.2 PCA function
eigenstrat <- function(geno){

  # Number of NA values for each SNPs
  nMis <- rowSums(is.na(geno))
  # Select the rows (SNPs) that do not have NA values
  #            (remove SNPs that have atleast one NA)
  geno <- geno[nMis == 0, ]      # 195 SNPs left

  # Sum of the individuals genotypes for each SNP / number individuals
  # (average genotype for each SNPs)
  avg <- rowSums(geno) / ncol(geno)
  # Remove the SNPs where all individuals are homozygous for a specific allele
  keep <- avg != 0 & avg != 2
  avg <- avg[keep]
  geno <- geno[keep,]

  # Number of SNPs after filtering (
  # removing homogeneous heter and SNPs with NA)
  snp <- nrow(geno)
  # Number of individual
  ind <- ncol(geno)

  # A allele freq (avg genotype / 2)
  freq <- avg / 2

  # Standardize the data (centeter and then normalize). 173 SPs (col) x 95 (rows)
  M <- (geno - avg) / sqrt(freq * (1 - freq))

  # I guess we find the eigenvectors and eigenvalues from cov matrix
  X <- t(M) %*% M # %*% is matrix moltiplication (transform M into a square matrix)
  X <- X / (sum(diag(X)) / (snp - 1))
  E <- eigen(X)
  mu <- (sqrt(snp - 1) + sqrt(ind)) ^ 2 / snp
  sigma <- (sqrt(snp - 1) + sqrt(ind)) / snp * (1 / sqrt(snp - 1) + 1 / sqrt(ind))
                                                                    ^ (1 / 3)
  E$TW <- (E$values[1] * ind / sum(E$values) - mu) / sigma
  E$mu <- mu
  E$sigma <- sigma
  class(E) <- "eigenstrat"
  return(E)
}
```

```
## 2.3 Coloring (wierd behaviour if code converted in a single function)
```

Plotting function

```
## 2.4 Plot
plot_PCA <- function(e,
                     title = "PCA",
                     xlab = "PC1",
                     ylab = "PC2",
                     legend_title = "",
                     leg_loc = "topright",
                     remove_outgroup = FALSE,
                     save = FALSE,
                     filename = "rplot.png"){
  # Set details
  colors = c("black", "black", "black", "black", "black")
  bg_point = c("limegreen", "orangered", "yellow", "deepskyblue1", "plum2")
  p_shape = c(21,21,21,21,25)
  individuals = c(37, 26, 10, 19, 3)
  species = c("N.g.granti", "N.g.robertsii", "N.notata", "N.petersii", "E.thomsonii")
  # Remove outgroup details if needed
  if (remove_outgroup == TRUE){
    colors = colors[-5]
    bg_point = bg_point[-5]
    p_shape = p_shape[-5]
    individuals = individuals[-5]
    species = species[-5]
  }
  if (save == TRUE){
  png(paste(filename, ".png"))
  }
  plot(e,
       col = "black",
       bg = rep(bg_point, individuals),
       cex = 1.2,
       panel.first = grid(lty = 1, lwd = 1, col = "darkgray"),
       xlab = xlab,
       ylab = ylab,
       pch = rep(p_shape, individuals),
       main = title,
       cex.main = 1.5
  )
  legend(leg_loc,
         legend = species,
         col = "black",
         pt.bg = bg_point,
         cex = 1.2,
         pch = p_shape,
         bg = "gray100",
         title = "Species")
  if (save == TRUE){
  dev.off()
```

```
  }
}
```

PCA 1 - Outgroup included, PC1 and PC2

```
#### PCA 1 - outgroup included, PC1 and PC2

# Preprocessing
geno1 <- preprocess(geno) # Sort individuals by species
                          # (doesn't remove outgroup by default)
geno1 <- detect_NA(geno1) # Transpose and set 0 values as NA

# Look at the principal components
summary(prcomp(na.omit(geno1))) # PC1: 74.9%
                                # PC2: 4.2%
                                # PC3: 3.3%

# Coloring (it looks like they created a eigenstrat class)
plot.eigenstrat <- function(x, col = 1, ...)   # ... Extendend slicing
  plot(x$vectors[, 1:2], col = col, ...)
print.eigenstrat <- function(x)
  cat("Statistic", x$TW, "n")
e <- eigenstrat(geno1)

# Plot
plot_PCA(e,
         title = "PCA by species (outgroup included)",
         xlab = "PC1 (74.9%)",
         ylab = "PC2 (4.2%)",
         save = FALSE,
         filename = "pca_outgroup_pc1_2")
```

PCA 2 - Outgroup included, PC2 and PC3

```
## PCA 2 - outgroup included, PC2 and PC3

# Preprocessing
geno2 <- preprocess(geno, remove_outgroup = FALSE)
geno2 <- detect_NA(geno2)

# Coloring
plot.eigenstrat <- function(x, col = 1, ...)
  plot(x$vectors[, 2:3], col = col, ...)
print.eigenstrat <- function(x)
  cat("Statistic", x$TW, "n")
e <- eigenstrat(geno2)

# Plot
plot_PCA(e,
         title = "PCA (outgroup included, pc2 and pc3)",
         xlab = "PC2 (4.2%)",
         ylab = "PC3 (3.3%)",
```

```
         leg_loc = "bottomright",
         save = FALSE,
         filename = "pca_outgroup_pc2_3")
```

PCA 3 - Outgroup excluded, PC1 and PC2

```
## PCA 3 - outgroup excluded, PC1 and PC2

# Preprocessing
geno3 <- preprocess(geno, remove_outgroup = TRUE)
geno3 <- detect_NA(geno3)

# Look at the principal components
summary(prcomp(na.omit(geno3))) # PC1: 77.3%
                                # PC2: 4.1%
                                # PC3: 2.1%
# Coloring
plot.eigenstrat <- function(x, col = 1, ...)
  plot(x$vectors[, 1:2], col = col, ...)
print.eigenstrat <- function(x)
  cat("Statistic", x$TW, "n")
e <- eigenstrat(geno3)

# Plot
plot_PCA(e,
         title = "PCA by species (outgroup excluded)",
         xlab = "PC1 (77.2%)",
         ylab = "PC2 (4.1%)",
         leg_loc = "bottomright",
         remove_outgroup = TRUE,
         save = FALSE,
         filename = "pca_NOoutgroup_pc1_2")
```

PCA 4 - Outgroup excluded, PC2 and PC3

```
## PCA 4 - outgroup excluded, PC2 and PC3

# Preprocessing
geno4 <- preprocess(geno, remove_outgroup = TRUE)
geno4 <- detect_NA(geno4)

# Coloring
plot.eigenstrat <- function(x, col = 1, ...)   # ... Extendend slicing
  plot(x$vectors[, 2:3], col = col, ...)
print.eigenstrat <- function(x)
  cat("Statistic", x$TW, "n")
e <- eigenstrat(geno4)

# Plot
plot_PCA(e,
         title = "PCA (outgroup excluded, pc2 and pc3)",
         xlab = "PC2 (4.1%)",
```

```
         ylab = "PC3 (2.1%)",
         leg_loc = "topleft",
         remove_outgroup = TRUE,
         save = FALSE,
         filename = "pca_NOoutgroup_pc2_3")
```

Plotting function with added localities information

```
## 2.4 Plot
plot_PCA_locality <- function(e,
                      title = "PCA",
                      xlab = "PC1",
                      ylab = "PC2",
                      legend_title = "",
                      leg_title = NULL,
                      leg_loc = "topright",
                      ncol = 1,
                      remove_outgroup = FALSE,
                      save = FALSE,
                      filename = "rplot.png"){
  # Set details

  colors = c("black", "black", "black", "black", "black")

  bg_point = c("lightgrey","grey36","yellow","deepskyblue1","limegreen",
               "orange","yellow","ivory","yellow","limegreen",
               "greenyellow","orangered","plum","orangered","navajowhite2",
               "cyan", "magenta","blue")

  leg_colors = c("lightgrey", "grey36","yellow","deepskyblue1","limegreen",
                 "orange","ivory", "greenyellow","orangered","plum",
                 "navajowhite2", "cyan", "magenta","blue")

  leg_colors2 = c("white", "white", "white", "white")

  leg_locality = c("Amboseli", "Nairobi", "Monduli", "Mkomazi West", "Mkomazi Est",
                   "Burigi", "Ikiri-Rungwa", "Masai Mara", "Maswa", "Ugalla",
                   "Samburu", "Sibiloi", "Tsavo", "Aruba Dam")

  leg_locality2 = c("N.g.granti", "N.g.robertsii", "N.notata", "N.petersii")

  p_shape = c(21,22,23,24)
  p_shape_leg = c(rep(21, 7), 22, 22, 22, 23, 23, 24, 24)
  p_shape_leg2 = c(1, 0, 5, 2)
  individuals_species = c(37, 26, 10, 19)
  individuals = c(3,5,3,4,6,1,7,1,5,2,11,12,1,2,7,3,11,8)

  locality = c("Amboseli", "Nairobi", "Monduli", "Mkomazi West", "Mkomazi Est",
               "Burigi",  "Monduli", "Ikiri-Rungwa", "Monduli", "Mkomazi Est",
               "Masai Mara", "Maswa", "Ugalla", "Maswa", "Samburu", "Sibiloi",
               "Tsavo", "Aruba Dam")
  # Output
  if (save == TRUE){
```

```
  png(paste(filename, ".png"))
  }
  plot(e,
       col = "black",
       bg = rep(bg_point, individuals),
       cex = 1.4,
       panel.first = grid(lty = 1, lwd = 1, col = "darkgray"),
       xlab = xlab,
       ylab = ylab,
       pch = rep(p_shape, individuals_species),
       main = title,
       cex.main = 1.5
  )
  legend(0.2182, -0.022,
         ncol = ncol,
         legend = leg_locality,
         col = "black",
         pt.bg = leg_colors,
         cex = 1,
         pch = p_shape_leg,
         bg = "gray100",
         title = leg_title)
   legend(0.4, 0.0699,
         legend = leg_locality2,
         col = "black",
         pt.bg = leg_colors2,
         cex = 1,
         pch = p_shape_leg2,
         bg = "gray100",
         title = "Species (shapes)")
  if (save == TRUE){
  dev.off()
  }
}
```

PCA 5 - Localities and species (Outgroup excluded)

```
## PCA 5 - Localities and species - Outgroup excluded, PC1 and PC2

# Preprocessing
geno5 <- preprocess(geno, remove_outgroup = TRUE)
geno5 <- detect_NA(geno3)

# Look at the principal components
summary(prcomp(na.omit(geno5))) # PC1: 77.3%
                                # PC2: 4.1%
                                # PC3: 2.1%
# Coloring
plot.eigenstrat <- function(x, col = 1, ...)
  plot(x$vectors[, 1:2], col = col, ...)
print.eigenstrat <- function(x)
  cat("Statistic", x$TW, "n")
e <- eigenstrat(geno3)
```

```
# Plot
plot_PCA_locality(e,
                  title = "PCA by localities and species (outgroup excluded)",
                  xlab = "PC1 (77.2%)",
                  ylab = "PC2 (4.1%)",
                  leg_title = "Localities (colors)",
                  ncol = 2,
                  remove_outgroup = TRUE,
                  save = FALSE,
                  filename = "pca_species_localities")
```

2. FIXATION INDEX (FST)

Weir and Cockerham Fst calculator

```
WC84 <- function(x, pop) {
  # Number individuals in each population
  n <- table(pop)
  # Number of populations
  npop <- nrow(n)
  # Average sample size of each population
  n_avg <- mean(n)
  # Total number of samples
  N <- length(pop)
  # Frequency in samples
  p <- apply(x, 2, function(x, pop) {
    tapply(x, pop, mean) / 2
  }, pop = pop)
  # Average frequency in all samples (apply(x,2,mean)/2)
  p_avg <- as.vector(n %*% p / N)
  # The sample variance of allele 1 over populations
  s2 <- 1 / (npop - 1) * (apply(p, 1, function(x) {
    ((x - p_avg) ^ 2)
  }) %*% n) / n_avg
  # Average heterozygotes
  #                         h <- apply(x == 1,2, function(x,pop)tapply(x,pop,mean),pop=pop)
  # Average heterozygote frequency for allele 1
  #                         h_avg <- as.vector(n %*% h / N)
  # Faster version than above:
  h_avg <- apply(x == 1, 2, sum) / N
  # Nc (see page 1360 in wier and cockerhamm, 1984)
  n_c <- 1 / (npop - 1) * (N - sum(n ^ 2) / N)
  # Variance betwen populations
  a <- n_avg / n_c * (s2 - (p_avg * (1 - p_avg) - (npop - 1) * s2 / npop - h_avg / 4) /
                      (n_avg - 1))
  # Variance between individuals
  b <- n_avg /
      (n_avg - 1) * (p_avg * (1 - p_avg) - (npop - 1) * s2 /
       npop - (2 * n_avg - 1) * h_avg / (4 * n_avg))
  # Variance within individuals
  c <- h_avg / 2
  # Inbreeding (F_it)
```

```
  F <- 1 - c / (a + b + c)
  # (F_st)
  theta <- a / (a + b + c)
  # (F_is)
  f <- 1 - c(b + c)
  # Weigted average of theta
  theta_w <- sum(a) / sum(a + b + c)
  list(F = F,
       theta = theta,
       f = f,
       theta_w = theta_w,
       a = a,
       b = b,
       c = c,
       total = c + b + a
  )
}
```

Perform Fst calculation between populations pairs

```
library(snpMatrix)

#### 1. Load data

## Retrieve information about species from popinfo file
popinfo = read.table("grants_popinfo.txt")
# To see the species represented in the data and how many individuals are there
table(popinfo[,2])
# Select species column
species = popinfo[2:nrow(popinfo),2]
popinfo_sorted = popinfo[order(popinfo[,2]),]

## Load genotype data (use all grants file bim, bed and fam)
data <- read.plink("grants")
geno <- matrix(as.integer(data@.Data), nrow = nrow(data@.Data))
dim(geno) # 95 rows (individuals),35174 columns (SNPs)


#### 2. Functions needed

## 2.1 Preprocessing

# Sort individuals by species and remove outgroup if needed
preprocess <- function(geno, remove_outgroup = FALSE){
  # Add species as last column
  geno <- cbind(geno, species)
  # Sort them by species
  geno <- geno[order(species),]
  if (remove_outgroup == TRUE){
    # Remove the outgroup
    geno <- geno[geno[,ncol(geno)] != 6,]
  }
  # Remove species column
```

```
  geno <- geno[,-ncol(geno)]
return(geno)
}

# Change values 0 to NA and change genotypes to 0, 1, 2
detect_NA <- function(geno){
  geno <- t(geno)
  geno[geno == 0] <- NA
  geno <- geno - 1
  return(geno)
}

#### 3. Preprocessing
geno <- preprocess(geno)
geno <- detect_NA(geno)

g <- geno[complete.cases(geno), ] # 189 x 95 (genotypes x individuals)
pop <- c(rep(1, 37), rep(2, 26), rep(3, 10), rep(4, 19), rep(5, 3))

#### 4. Population comparisons

# List of population labels for comparison
pop12 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 2, TRUE, FALSE))]
pop13 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 3, TRUE, FALSE))]
pop14 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 4, TRUE, FALSE))]
pop15 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop23 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 3, TRUE, FALSE))]
pop24 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 4, TRUE, FALSE))]
pop25 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop34 <- pop[ifelse(pop == 3, TRUE, ifelse(pop == 4, TRUE, FALSE))]
pop35 <- pop[ifelse(pop == 3, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop45 <- pop[ifelse(pop == 4, TRUE, ifelse(pop == 5, TRUE, FALSE))]
# Genotypes of the compared populations
g12 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 2, TRUE, FALSE))]
g13 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 3, TRUE, FALSE))]
g14 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 4, TRUE, FALSE))]
g15 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g23 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 3, TRUE, FALSE))]
g24 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 4, TRUE, FALSE))]
g25 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g34 <- g[, ifelse(pop == 3, TRUE, ifelse(pop == 4, TRUE, FALSE))]
g35 <- g[, ifelse(pop == 3, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g45 <- g[, ifelse(pop == 4, TRUE, ifelse(pop == 5, TRUE, FALSE))]
# Run Fst calculator for the different populations comparison (return value for each SNP)
result12 <- WC84(t(g12), pop12)
result13 <- WC84(t(g13), pop13)
result14 <- WC84(t(g14), pop14)
result15 <- WC84(t(g15), pop15)
result23 <- WC84(t(g23), pop23)
result24 <- WC84(t(g24), pop24)
result25 <- WC84(t(g25), pop25)
result34 <- WC84(t(g34), pop34)
result35 <- WC84(t(g35), pop35)
```

```
result45 <- WC84(t(g45), pop45)
# Compute the mean Fst between SNPs for each population comparison
Fst12 <- mean(result12$theta, na.rm = T)
Fst13 <- mean(result13$theta, na.rm = T)
Fst14 <- mean(result14$theta, na.rm = T)
Fst15 <- mean(result15$theta, na.rm = T)
Fst23 <- mean(result23$theta, na.rm = T)
Fst24 <- mean(result24$theta, na.rm = T)
Fst25 <- mean(result25$theta, na.rm = T)
Fst34 <- mean(result34$theta, na.rm = T)
Fst35 <- mean(result35$theta, na.rm = T)
Fst45 <- mean(result45$theta, na.rm = T)
```

Print the output

```
# Output the results
all_results <- c(Fst12,Fst13,Fst14,Fst15,Fst23,
                 Fst24,Fst25,Fst34,Fst35,Fst45)

results_names <- c("Pop 1-2   (N.G.grantii - N.G.robertsii)",
                   "Pop 1-3   (N.G.grantii - N.notata)",
                   "Pop 1-4   (N.G.grantii - N.petersii)",
                   "Pop 1-5   (N.G.grantii - E.thomsonii)",
                   "Pop 2-3 (N.G.robertsii - N.notata)",
                   "Pop 2-4 (N.G.robertsii - N.petersii)",
                   "Pop 2-5 (N.G.robertsii - E.thomsonii)",
                   "Pop 3-4      (N.notata - N.petersii)",
                   "Pop 3-5      (N.notata - E.thomsonii)",
                   "Pop 4-5    (N.petersii - E.thomsonii)")

test <- array(all_results[order(all_results)],
              dimnames = list(results_names[order(all_results)],
                              c("Fst coefficients")),
              dim = c(10,1))
test
```

Perform Fst calculation between populations pairs

(Mkomazi placed as individual population, remember to run the WC84 function)

```
library(snpMatrix)

#### 1. Load data

## Retrieve information about species from popinfo file
popinfo = read.table("grants_popinfo.txt")
# To see the species represented in the data and how many individuals are there
table(popinfo[,2])
# Select species column
species = popinfo[2:nrow(popinfo),2]
popinfo_sorted = popinfo[order(popinfo[,2]),]

## Load genotype data (use all grants file bim, bed and fam)
data <- read.plink("grants")
```

```r
geno <- matrix(as.integer(data@.Data), nrow = nrow(data@.Data))
dim(geno) # 95 rows (individuals),35174 columns (SNPs)


#### 2. Functions needed

## 2.1 Preprocessing

# Sort individuals by species and remove outgroup if needed
preprocess <- function(geno, remove_outgroup = FALSE){
  # Add species as last column
  geno <- cbind(geno, species)
  # Sort them by species
  geno <- geno[order(species),]
  if (remove_outgroup == TRUE){
    # Remove the outgroup
    geno <- geno[geno[,ncol(geno)] != 6,]
  }
  # Remove species column
  geno <- geno[,-ncol(geno)]
return(geno)
}

# Change values 0 to NA and change genotypes to 0, 1, 2
detect_NA <- function(geno){
  geno <- t(geno)
  geno[geno == 0] <- NA
  geno <- geno - 1
  return(geno)
}

#### 3. Preprocessing
geno <- preprocess(geno)
geno <- detect_NA(geno)

g <- geno[complete.cases(geno), ] # 189 x 95 (genotypes x individuals)
                                  # granti 1-11   mkomazi 12-21 + 36,37
pop <- c(rep(1, 37), rep(2, 26), rep(3, 10), rep(4, 19), rep(5, 3))
pop <- c(rep(1, 11), rep(6, 10), rep(1, 14), rep(6, 2), rep(2, 26),
         rep(3, 10), rep(4, 19), rep(5, 3))

#### 4. Population comparisons

# List of population labels for comparison
pop12 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 2, TRUE, FALSE))]
pop13 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 3, TRUE, FALSE))]
pop14 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 4, TRUE, FALSE))]
pop15 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop16 <- pop[ifelse(pop == 1, TRUE, ifelse(pop == 6, TRUE, FALSE))]
pop23 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 3, TRUE, FALSE))]
pop24 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 4, TRUE, FALSE))]
pop25 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop26 <- pop[ifelse(pop == 2, TRUE, ifelse(pop == 6, TRUE, FALSE))]
```

```
pop34 <- pop[ifelse(pop == 3, TRUE, ifelse(pop == 4, TRUE, FALSE))]
pop35 <- pop[ifelse(pop == 3, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop36 <- pop[ifelse(pop == 3, TRUE, ifelse(pop == 6, TRUE, FALSE))]
pop45 <- pop[ifelse(pop == 4, TRUE, ifelse(pop == 5, TRUE, FALSE))]
pop46 <- pop[ifelse(pop == 4, TRUE, ifelse(pop == 6, TRUE, FALSE))]
pop56 <- pop[ifelse(pop == 5, TRUE, ifelse(pop == 6, TRUE, FALSE))]
# Genotypes of the compared populations
g12 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 2, TRUE, FALSE))]
g13 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 3, TRUE, FALSE))]
g14 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 4, TRUE, FALSE))]
g15 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g16 <- g[, ifelse(pop == 1, TRUE, ifelse(pop == 6, TRUE, FALSE))]
g23 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 3, TRUE, FALSE))]
g24 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 4, TRUE, FALSE))]
g25 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g26 <- g[, ifelse(pop == 2, TRUE, ifelse(pop == 6, TRUE, FALSE))]
g34 <- g[, ifelse(pop == 3, TRUE, ifelse(pop == 4, TRUE, FALSE))]
g35 <- g[, ifelse(pop == 3, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g36 <- g[, ifelse(pop == 3, TRUE, ifelse(pop == 6, TRUE, FALSE))]
g45 <- g[, ifelse(pop == 4, TRUE, ifelse(pop == 5, TRUE, FALSE))]
g46 <- g[, ifelse(pop == 4, TRUE, ifelse(pop == 6, TRUE, FALSE))]
g56 <- g[, ifelse(pop == 5, TRUE, ifelse(pop == 6, TRUE, FALSE))]
# Run Fst calculator for the different populations comparison (return value for each SNP)
result12 <- WC84(t(g12), pop12)
result13 <- WC84(t(g13), pop13)
result14 <- WC84(t(g14), pop14)
result15 <- WC84(t(g15), pop15)
result16 <- WC84(t(g16), pop16)
result23 <- WC84(t(g23), pop23)
result24 <- WC84(t(g24), pop24)
result25 <- WC84(t(g25), pop25)
result26 <- WC84(t(g26), pop26)
result34 <- WC84(t(g34), pop34)
result35 <- WC84(t(g35), pop35)
result36 <- WC84(t(g36), pop36)
result45 <- WC84(t(g45), pop45)
result46 <- WC84(t(g46), pop46)
result56 <- WC84(t(g56), pop56)
# Compute the mean Fst between SNPs for each population comparison
Fst12 <- mean(result12$theta, na.rm = T)
Fst13 <- mean(result13$theta, na.rm = T)
Fst14 <- mean(result14$theta, na.rm = T)
Fst15 <- mean(result15$theta, na.rm = T)
Fst16 <- mean(result16$theta, na.rm = T)
Fst23 <- mean(result23$theta, na.rm = T)
Fst24 <- mean(result24$theta, na.rm = T)
Fst25 <- mean(result25$theta, na.rm = T)
Fst26 <- mean(result26$theta, na.rm = T)
Fst34 <- mean(result34$theta, na.rm = T)
Fst35 <- mean(result35$theta, na.rm = T)
Fst36 <- mean(result36$theta, na.rm = T)
Fst45 <- mean(result45$theta, na.rm = T)
Fst46 <- mean(result46$theta, na.rm = T)
```

```
Fst56 <- mean(result56$theta, na.rm = T)
```

Print the output

```
# Output the results
all_results <- c(Fst12,Fst13,Fst14,Fst15,Fst23,
                 Fst24,Fst25,Fst34,Fst35,Fst45,
                 Fst16,Fst26,Fst36,Fst46, Fst56)

results_names <- c("Pop 1-2   (N.g.grantii - N.g.robertsii)",
                   "Pop 1-3   (N.g.grantii - N.notata)",
                   "Pop 1-4   (N.g.grantii - N.petersii)",
                   "Pop 1-5   (N.g.grantii - E.thomsonii)",
                   "Pop 2-3 (N.g.robertsii - N.notata)",
                   "Pop 2-4 (N.g.robertsii - N.petersii)",
                   "Pop 2-5 (N.g.robertsii - E.thomsonii)",
                   "Pop 3-4      (N.notata - N.petersii)",
                   "Pop 3-5      (N.notata - E.thomsonii)",
                   "Pop 4-5    (N.petersii - E.thomsonii)",
                   "Pop 1-6    (N.g.granti - Mkomazi)",
                   "Pop 2-6    (N.g.robertsii - Mkomazi)",
                   "Pop 3-6    (N.notata - Mkomazi)",
                   "Pop 4-6    (N.petersii - Mkomazi)",
                   "Pop 5-6    (E.thomsonii - Mkomazi")

test <- array(all_results[order(all_results)],
              dimnames = list(results_names[order(all_results)],
                              c("Fst coefficients")),
              dim = c(length(all_results),1))
test
```

2.6 Fst calculation by location

(removed localities and samples that could create bias)

```
library(snpMatrix)

#### 1. Load data

## Retrieve information about species from popinfo file
popinfo = read.table("grants_popinfo.txt")[-1,]

## Load genotype data (use all grants file bim, bed and fam)
data <- read.plink("grants")
geno <- matrix(as.integer(data@.Data), nrow = nrow(data@.Data))
dim(geno) # 95 rows (individuals),35174 columns (SNPs)


#############
# From other analysis I believe that this is a sample mistake
  # Remove 1
granti = popinfo[popinfo[,2] == "g.granti",]
ikiri = granti[granti[1] == "Ikiri-Rungwa"]
order_ikiri = ikiri[4]
```

```
  # Remove 2
burigi = popinfo[,1] == "Burigi"
order_burigi = popinfo[burigi,][4]
  # Remove 3
ugalla = popinfo[,1] == "Ugalla"
order_ugalla = popinfo[ugalla,][4]
  # Remove them from geno
geno = geno[c(-68, -57, -91),]
popinfo = popinfo[c(-68, -57, -91),]

#############

popinfo = popinfo[order(popinfo[,1]),]
species = popinfo[,2]
localities = popinfo[,1]

#### 2. Functions needed for Fst

## 2.1 Preprocessing

# Sort dataset by localities
geno = geno[order(localities),]

# Change values 0 to NA and change genotypes to 0, 1, 2
detect_NA <- function(geno){
  geno <- t(geno)
  geno[geno == 0] <- NA
  geno <- geno - 1
  return(geno)
}
geno <- detect_NA(geno)

g <- geno[complete.cases(geno), ] # 189 x 95 (genotypes x individuals)
# We consider 13 populations (mkomazi grouped together)
pop <- c(rep(1, 3), rep(2, 8), rep(3, 3), rep(4, 11),
         rep(5, 14), rep(6,12), rep(7, 15), rep(8, 5),
         rep(9, 7), rep(10, 3), rep(11, 11))

pop_names <- c("Amboseli", "Aruba_Dam", "Ikiri-Rungwa", "Masai_Mara",
               "Maswa", "Mkomazi", "Monduli", "Nairobi", "Samburu", "Sibiloi",
               "Tsavo")

pop_names_species <- c("granti", "robertsii", "notata", "petersi", "thomsoni")

# I should remove the Burigi and Ugalla sample

#### 4. Population comparisons
fst_comparison <- function(nclusters){
  ## Get the labels for the populations comparison
  index = 1
  poplist_ij = list()
  poplistpair = list()
  nameslistlabel1 = list()
```

```
nameslistlabel2 = list()
for (label1 in 1:nclusters){
  for (label2 in 2:nclusters){
    if (label1 < label2){
     # Save labels
     poplist_ij[[index]] =
       pop[ifelse(pop == label1, TRUE, ifelse(pop == label2, TRUE, FALSE))]
     # Save the population comparison pair index
     poplistpair[index] = paste("pop", label1, label2, sep="_")
     # Save the population comparison names
     nameslistlabel1[index] = pop_names[label1]
     nameslistlabel2[index] = pop_names[label2]
     index = index + 1
   }
  }
}
## Genotypes of the compared populations
index = 1
glist_ij = list()
for (label1 in 1:nclusters){
  for (label2 in 2:nclusters){
    if (label1 < label2){
     # Save labels
     glist_ij[[index]] =
       g[, ifelse(pop == label1, TRUE, ifelse(pop == label2, TRUE, FALSE))]
     # Save the population comparison pair
     index = index + 1
   }
  }
}

# Run Fst calculator for the different populations comparison (return value for each SNP)
index = 1
resultlist_ij = list()
for (label1 in 1:nclusters){
  for (label2 in 2:nclusters){
    if (label1 < label2){
     resultlist_ij[[index]] = WC84(t(glist_ij[[index]]), poplist_ij[[index]])
     index = index + 1
   }
  }
}

# Compute the mean Fst between SNPs for each population comparison
index = 1
fstlist_ij = list()
for (label1 in 1:nclusters){
  for (label2 in 2:nclusters){
    if (label1 < label2){
     fstlist_ij[[index]] = mean(resultlist_ij[[index]]$theta, na.rm = T)
     index = index + 1
   }
  }
```

```
  }

  # Create an array with the output of the Fst analysis
  index = 1
  fstarray = array(dim = c(length(poplistpair),2))
  for (fst in fstlist_ij){
    fstarray[index,1] = poplistpair[[index]]
    fstarray[index,2] = fst
    index = index + 1
  }
  # Sort the array by Fst
  label1_sorted = nameslistlabel1[order(fstarray[,2])]
  label2_sorted = nameslistlabel2[order(fstarray[,2])]
  fstarray_sorted = fstarray[order(fstarray[,2]),]
  # Combine the names of the population of the pairs to the array
  fstarray_sorted = cbind(fstarray_sorted, label1_sorted)
  fstarray_sorted = cbind(fstarray_sorted, label2_sorted)
  fstarray = cbind(fstarray, nameslistlabel1)
  fstarray = cbind(fstarray, nameslistlabel2)
  # Output two txt files
  write.table(fstarray,
              file = "fst_bylocalities.txt", sep="\t",
              col.names = F, row.names = F)
  write.table(fstarray_sorted,
              file = "fst_bylocalities_sorted.txt", sep="\t",
              col.names = F, row.names = F)
  return(fstarray)
}

fstarray = fst_comparison(11)

mkomazi = fstarray[fstarray[,3]=="Mkomazi" | fstarray[,4] == "Mkomazi",]
fstarray[fstarray[,3]=="Sibiloi" | fstarray[,4] == "Sibiloi",]
```

3. ADMIXTURE

We divide the data in k populations to test alternative divisions (bash terminal)

```
# Generate .P and .Q files for admixture analysis
for i in 3 4 5 6 7 8 9 10; do admixture --cv grants.bed $i; done > cvoutput
# Print all CV error for each pop subdivision
grep -i 'CV error' cvoutput
```

CV error (K=3): 0.16170
CV error (K=4): 0.15732
CV error (K=5): 0.15123
CV error (K=6): 0.14432
CV error (K=7): 0.14160
CV error (K=8): 0.15132

```
# Run admixture (Fst generated between estimated population, k = 7)
admixture grants.bed 7
# Look at results (not really useful)
```

```
less -S grants.7.Q
```

Plot admixture

```
library(snpMatrix)

## 1. Load data and preprocessing

## Retrieve information about species from popinfo file
popinfo = read.table("grants_popinfo.txt")
# Select species column
species = popinfo[2:nrow(popinfo),2]

## Load genotype data (use all grants file bim, bed and fam)
data <- read.plink("grants")
geno <- matrix(as.integer(data@.Data), nrow = nrow(data@.Data))
dim(geno) # 95 rows (individuals),35174 columns (SNPs)

# Simple function to sort individuals
sort_by <- function(data, by){
  return(data[order(by),])
}
# Sort the popinfo file to add information to admixture plot
info_sorted <- popinfo[2:nrow(popinfo),]
info_sorted <- sort_by(info_sorted, species)

# Load (and sort) admixture data with different k used (populations divisions)
snpk3 = read.table("grants.3.Q")
snpk3 <- sort_by(snpk3, species)
snpk4 = read.table("grants.4.Q")
snpk4 <- sort_by(snpk4, species)
snpk5 = read.table("grants.5.Q")
snpk5 <- sort_by(snpk5, species)
snpk6 = read.table("grants.6.Q")
snpk6 <- sort_by(snpk6, species)
snpk7 = read.table("grants.7.Q")
snpk7 <- sort_by(snpk7, species)
snpk8 = read.table("grants.8.Q")
snpk8 <- sort_by(snpk8, species)

## 3. Plot
species = info_sorted[,2]
locality = info_sorted[,1]
# Change locality names
levels(locality)[4] = "Ikiri-Rungwa"
levels(locality)[2] = "Aruba Dam"
levels(locality)[6] = "Masai Mara"
levels(locality)[8] = "Mkomazi Est"
levels(locality)[9] = "Mkomazi West"

## First three population division by ancestry
par(mfrow = c(4, 1))
```

```
# 3 ancestral populations
par(fig=c(0, 1, 0.6, 0.85))
barplot(t(as.matrix(snpk3)),
        col = c("pink", "deepskyblue1", "limegreen"),
        border = "black",
        names.arg = rep("", 95),
  #      horiz = FALSE,
        las = 2,
        ylab = "K = 3"
)
par(las=2)
axis(3, seq(0.5, 113.4,1.2), tck=FALSE, labels = locality)
# 4 ancestral populations
par(fig=c(0, 1, 0.5, 0.75), new=TRUE)
barplot(t(as.matrix(snpk4)),
        col = c("pink", "tomato1", "limegreen", "deepskyblue1"),
        border = "black",
        names.arg = rep("", 95),
        las = 2,
        ylab = "K = 4"
)
# 5 ancestral populations
par(fig=c(0, 1, 0.4, 0.65), new=TRUE)
barplot(t(as.matrix(snpk5)),
        col = c("pink", "limegreen", "deepskyblue1", "darkgoldenrod1", "tomato1"),
        border = "black",
        names.arg = rep("", 95),
        cex.names = 0.8,
        las = 2,
        ylab = "K = 5"
)
# 6 ancestral populations
par(fig=c(0, 1, 0.3, 0.55), new=TRUE)
barplot(t(as.matrix(snpk6)),
        col = c("pink", "deepskyblue1", "limegreen",
                "orchid", "darkgoldenrod1", "tomato1"),
        border = "black",
        names.arg = rep("", 95),
        cex.names = 0.8,
        las = 2,
        ylab = "K = 6"
)
# 7 ancestral populations
par(fig=c(0, 1, 0.2, 0.45), new=TRUE)
barplot(t(as.matrix(snpk7)),
        col = c("limegreen", "orchid", "deepskyblue1",
                "tomato1", "pink", "ivory2", "darkgoldenrod1"),
        border = "black",
        names.arg = rep("", 95),
        cex.names = 0.8,
        las = 2,
        ylab = "K = 7"
)
```

```
# 8 ancestral populations
par(fig=c(0, 1, 0.1, 0.35), new=TRUE)
barplot(t(as.matrix(snpk8)),
        col = c("navajowhite2", "tomato1", "darkgoldenrod1",
                "deepskyblue1", "limegreen", "orchid", "pink", "ivory2"),
        border = "black",
        names.arg = species,
        cex.names = 1,
        las = 2,
        ylab = "K = 8"
)
```

4 GENETIC DIVERSITY (and bed fam files generation)

4.1 Genetic diversity by species

Step 1

We need a txt file composed of two column

- The indixes of the samples we want to keep
- A family ID column that in our case will be filled with 1

```
### This script is for making id files. They specify which individiuals
### that are kept in the new bim/bed/fam files.

#Here i read in the species information.
popinfo <- read.table("grants_popinfo.txt", header = T)

#Here I make a within-family id column(just a column of ones in our case)
popinfo$wf_id <- 1

new_id <- read.table("grants.fam")
popinfo$Order <- new_id$V1

#Here I subset table for each species
granti  <- subset(popinfo, popinfo$Species == "g.granti")
robertsii  <- subset(popinfo, popinfo$Species == "g.robertsii")
granti_mod <- subset(granti, granti$Locality != "Mkomazi_E")
granti_mod <- subset(granti_mod, granti_mod$Locality != "Mkomazi_W")
notata  <- subset(popinfo, popinfo$Species == "notata")
thomsons  <- subset(popinfo, popinfo$Species == "thomsons")
petersii  <- subset(popinfo, popinfo$Species == "petersii")

#Here I keep the columns needed for subsetting in plink
granti  <- granti[, c("Order", "wf_id")]
granti_mod  <- granti_mod[, c("Order", "wf_id")]
robertsii  <- robertsii[, c("Order", "wf_id")]
notata  <- notata[, c("Order", "wf_id")]
thomsons  <- thomsons[, c("Order", "wf_id")]
petersii  <- petersii[, c("Order", "wf_id")]
robertsii_and_granti <- rbind(granti, robertsii)
robertsii_and_granti_NOmko <- rbind(granti_mod, robertsii)
```

```r
no_mkomazi <- rbind(granti_mod, robertsii, notata, thomsons, petersii)

#Here I save all the files
write.table(granti,
            file = "granti_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(granti_mod,
            file = "granti_mod_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(robertsii,
            file = "robertsii_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(notata,
            file = "notata_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(thomsons,
            file = "thomsons_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(petersii,
            file = "petersii_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(robertsii_and_granti,
            file = "robertsii_and_granti_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)

write.table(no_mkomazi,
            file = "no_mkomazi_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
```

```
write.table(robertsii_and_granti_NOmko,
            file = "robertsii_and_granti_NOmko_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
```

Step 2

We want to create bed and fam files for clusters (species)

```
### Create bed and fam files
plink --bfile grants --make-bed --keep granti_id.txt --out granti
plink --bfile grants --make-bed --keep robertsii_id.txt --out robertsii
plink --bfile grants --make-bed --keep notata_id.txt --out notata
plink --bfile grants --make-bed --keep thomsons_id.txt --out thomsons
plink --bfile grants --make-bed --keep petersii_id.txt --out petersii
plink --bfile grants --make-bed --keep granti_mod_id.txt --out granti_mod
plink --bfile grants --make-bed --keep robertsii_and_granti_id.txt
--out robertsii_and_granti
plink --bfile grants --make-bed --keep no_mkomazi_id.txt --out no_mkomazi
plink --bfile grants --make-bed --keep robertsii_and_granti_NOmko_id.txt
--out robertsii_and_granti_NOmko
```

Step 3

We now want to create an allele frequency file for each cluster (species).

We are calculating the allele frequency of the minor allele for each variable site.

```
# Create allele frequency files
plink --noweb --bfile granti --freq --out granti
plink --noweb --bfile granti_mod --freq --out granti_mod
plink --noweb --bfile notata --freq --out notata
plink --noweb --bfile robertsii --freq --out robertsii
plink --noweb --bfile petersii --freq --out petersii
plink --noweb --bfile thomsons --freq --out thomsons
plink --noweb --bfile robertsii_and_granti --freq --out robertsii_and_granti
plink --noweb --bfile robertsii_and_granti_NOmko --freq --out robertsii_and_granti_NOmko

# Remove NA
cat granti.frq |grep -v NA > granti_noNA.frq
cat granti_mod.frq |grep -v NA > granti_mod_noNA.frq
cat notata.frq |grep -v NA > notata_noNA.frq
cat robertsii.frq |grep -v NA > robertsii_noNA.frq
cat petersii.frq |grep -v NA > petersii_noNA.frq
cat thomsons.frq |grep -v NA > thomsons_noNA.frq
cat robertsii_and_granti.frq |grep -v NA > robertsii_and_granti_noNA.frq
cat robertsii_and_granti_NOmko.frq |grep -v NA > robertsii_and_granti_NOmko_noNA.frq
```

Step 4

We now want to calculate the genetic diversity within clusters (species in this case)

```
# Read in each of the frequency files
granti<-read.table("granti_noNA.frq",h=T)
granti_mod<-read.table("granti_mod_noNA.frq",h=T)
notata<-read.table("notata_noNA.frq",h=T)
robertsii<-read.table("robertsii_noNA.frq",h=T)
petersii<-read.table("petersii_noNA.frq",h=T)
thomsons<-read.table("thomsons_noNA.frq",h=T)
robertsii_and_granti<-read.table("robertsii_and_granti_noNA.frq",h=T)
robertsii_and_granti_NOmko <- read.table("robertsii_and_granti_NOmko_noNA.frq", h = T)

# Function for estimating the expected heterozygosity
het<-function(x){2*x*(1-x)}

# Remove all fixed alleles in each population
thomsons <- thomsons[thomsons[,"MAF"]>0,]
notata <- notata[notata[,"MAF"]>0,]
robertsii <- robertsii[robertsii[,"MAF"]>0,]
petersii <- petersii[petersii[,"MAF"]>0,]
granti <- granti[granti[,"MAF"]>0,]
granti_mod <- granti_mod[granti_mod[,"MAF"]>0,]
robertsii_and_granti <- robertsii_and_granti[robertsii_and_granti[,"MAF"]>0,]
robertsii_and_granti_NOmko <-
  robertsii_and_granti_NOmko[robertsii_and_granti_NOmko[,"MAF"] > 0,]

total_sites <- 5909791 # from email by Genis

# The pi-values for each polymorphic SNP
thomsons <- cbind(thomsons, pi=het(thomsons$MAF) *(length(thomsons$MAF)/(total_sites)))
notata <- cbind(notata, pi=het(notata$MAF) *(length(notata$MAF)/(total_sites)))
robertsii <- cbind(robertsii, pi=het(robertsii$MAF) *(length(robertsii$MAF)/
                                                        (total_sites)))
petersii <- cbind(petersii, pi=het(petersii$MAF) *(length(petersii$MAF)/(total_sites)))
granti <- cbind(granti, pi=het(granti$MAF) *(length(granti$MAF)/(total_sites)))
granti_mod <- cbind(granti_mod, pi=het(granti_mod$MAF) *(length(granti_mod$MAF)/
                                                           (total_sites)))
robertsii_and_granti <- cbind(robertsii_and_granti, pi=het(robertsii_and_granti$MAF)
                              *(length(robertsii_and_granti$MAF)/(total_sites)))
robertsii_and_granti_NOmko <-
  cbind(robertsii_and_granti_NOmko, pi = het(robertsii_and_granti_NOmko$MAF)
  *(length(robertsii_and_granti_NOmko$MAF) / (total_sites)))

png('nucleotide_diversity_by_species.png')
par(mfrow = c(1, 1), cex.axis=0.6)
val = c(
  mean(granti$pi),
  mean(granti_mod$pi),
  mean(robertsii$pi),
  mean(robertsii_and_granti$pi),
  mean(robertsii_and_granti_NOmko$pi),
  mean(petersii$pi),
```

```
  mean(notata$pi),
  mean(thomsons$pi)
)
names = c(
    "granti",
    "gr. (no mko)",
    "robertsii",
    "rob + gr",
    "rob + gr (no mko)",
    "petersii",
    "notata",
    "thomsonii")
barplot(
  val,
  ylim = c(0.000, 0.0005),
  ylab = "pi",
  xlab = "Population",
  main = "Genetic differences by species and subspecies",
  col = c("limegreen","green","red","blueviolet",
          "mediumpurple1","deepskyblue",
          "yellow","pink"),
  names.arg = names
)
dev.off()

pi_array <- array(val, dimnames = list(names, c("Pi")),
                  dim = c(8,1))
pi_array
```

4.2 Genetic diversity by locations

(repeat all steps for populations divided by locations)

Step 2.1

We need a txt file composed of two column

- The indixes of the samples we want to keep
- A family ID column that in our case will be filled with 1

```
# Here I read in the species information.
popinfo <- read.table("grants_popinfo.txt", header = T)
table(popinfo[,2])

# Here I make a within-family id column(just a column of ones in our case)
popinfo$wf_id <- 1

new_id <- read.table("grants.fam")
popinfo$Order <- new_id$V1

# Here I subset table for each locality
Amboseli <- subset(popinfo, popinfo$Locality == "Amboseli")
Aruba_Dam <- subset(popinfo, popinfo$Locality == "Aruba_Dam")
Ikiri_Rungwa <- subset(popinfo, popinfo$Locality == "Ikiri-Rungwa"
```

```
                      & popinfo$Species == "thomsons")
Masai_Mara <- subset(popinfo, popinfo$Locality == "Masai_Mara")
Maswa <- subset(popinfo, popinfo$Locality == "Maswa")
Mkomazi <- subset(popinfo, popinfo$Locality == "Mkomazi_E"
                  | popinfo$Locality == "Mkomazi_W")
Monduli <- subset(popinfo, popinfo$Locality == "Monduli")
Nairobi <- subset(popinfo, popinfo$Locality == "Nairobi")
Samburu <- subset(popinfo, popinfo$Locality == "Samburu")
Sibiloi <- subset(popinfo, popinfo$Locality == "Sibiloi")
Tsavo <- subset(popinfo, popinfo$Locality == "Tsavo")

# Here I keep the columns needed for subsetting in plink
Amboseli <- Amboseli[, c("Order", "wf_id")]
Aruba_Dam <- Aruba_Dam[, c("Order", "wf_id")]
Ikiri_Rungwa <- Ikiri_Rungwa[, c("Order", "wf_id")]
Masai_Mara <- Masai_Mara[, c("Order", "wf_id")]
Maswa <- Maswa[, c("Order", "wf_id")]
Mkomazi <- Mkomazi[, c("Order", "wf_id")]
Monduli <- Monduli[, c("Order", "wf_id")]
Nairobi <- Nairobi[, c("Order", "wf_id")]
Samburu <- Samburu[, c("Order", "wf_id")]
Sibiloi <- Sibiloi[, c("Order", "wf_id")]
Tsavo <- Tsavo[, c("Order", "wf_id")]

# Here I save all the files
write.table(Amboseli,
            file = "Amboseli_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Aruba_Dam,
            file = "Aruba_Dam_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Ikiri_Rungwa,
            file = "Ikiri_Rungwa_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Masai_Mara,
            file = "Masai_Mara_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Maswa,
            file = "Maswa_id.txt",
            sep = "\t",
            col.names = F,
```

```
            row.names = F
)
write.table(Mkomazi,
            file = "Mkomazi_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Monduli,
            file = "Monduli_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Nairobi,
            file = "Nairobi_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Samburu,
            file = "Samburu_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Samburu,
            file = "Samburu_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Sibiloi,
            file = "Sibiloi_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
write.table(Tsavo,
            file = "Tsavo_id.txt",
            sep = "\t",
            col.names = F,
            row.names = F
)
```

Step 2

We want to create bed and fam files for clusters (species)

```
# Create bed and fam files
plink --bfile grants --make-bed --keep Amboseli_id.txt --out Amboseli
plink --bfile grants --make-bed --keep Aruba_Dam_id.txt --out Aruba_Dam
plink --bfile grants --make-bed --keep Ikiri_Rungwa_id.txt --out Ikiri_Rungwa
```

```
plink --bfile grants --make-bed --keep Masai_Mara_id.txt --out Masai_Mara
plink --bfile grants --make-bed --keep Maswa_id.txt --out Maswa
plink --bfile grants --make-bed --keep Mkomazi_id.txt --out Mkomazi
plink --bfile grants --make-bed --keep Monduli_id.txt --out Monduli
plink --bfile grants --make-bed --keep Nairobi_id.txt --out Nairobi
plink --bfile grants --make-bed --keep Samburu_id.txt --out Samburu
plink --bfile grants --make-bed --keep Sibiloi_id.txt --out Sibiloi
plink --bfile grants --make-bed --keep Tsavo_id.txt --out Tsavo
```

Step 3

We now want to create an allele frequency file for each cluster (species).

We are calculating the allele frequency of the minor allele for each variable site.

```
# Create files with minor allele frequency for each SNP for each cluster
plink --noweb --bfile Amboseli --freq --out Amboseli
plink --noweb --bfile Aruba_Dam --freq --out Aruba_Dam
plink --noweb --bfile Ikiri_Rungwa --freq --out Ikiri_Rungwa
plink --noweb --bfile Masai_Mara --freq --out Masai_Mara
plink --noweb --bfile Maswa --freq --out Maswa
plink --noweb --bfile Mkomazi --freq --out Mkomazi
plink --noweb --bfile Monduli --freq --out Monduli
plink --noweb --bfile Nairobi --freq --out Nairobi
plink --noweb --bfile Samburu --freq --out Samburu
plink --noweb --bfile Sibiloi --freq --out Sibiloi
plink --noweb --bfile Tsavo --freq --out Tsavo

# Remove NA
cat Amboseli.frq |grep -v NA > Amboseli_noNA.frq
cat Aruba_Dam.frq |grep -v NA > Aruba_Dam_noNA.frq
cat Ikiri_Rungwa.frq |grep -v NA > Ikiri_Rungwa_noNA.frq
cat Masai_Mara.frq |grep -v NA > Masai_Mara_noNA.frq
cat Maswa.frq |grep -v NA > Maswa_noNA.frq
cat Mkomazi.frq |grep -v NA > Mkomazi_noNA.frq
cat Monduli.frq |grep -v NA > Monduli_noNA.frq
cat Nairobi.frq |grep -v NA > Nairobi_noNA.frq
cat Samburu.frq |grep -v NA > Samburu_noNA.frq
cat Sibiloi.frq |grep -v NA > Sibiloi_noNA.frq
cat Tsavo.frq |grep -v NA > Tsavo_noNA.frq
```

Step 4

We now want to calculate the genetic diversity within clusters (species)

```
# Read in each of the frequency files
Amboseli <- read.table("Amboseli_noNA.frq", h = T)
Aruba_Dam <- read.table("Aruba_Dam_noNA.frq", h = T)
Ikiri_Rungwa <- read.table("Ikiri_Rungwa_noNA.frq", h = T)
Masai_Mara <- read.table("Masai_Mara_noNA.frq", h = T)
Maswa <- read.table("Maswa_noNA.frq", h = T)
Mkomazi <- read.table("Mkomazi_noNA.frq", h = T)
Monduli <- read.table("Monduli_noNA.frq", h = T)
Nairobi <- read.table("Nairobi_noNA.frq", h = T)
```

```
Samburu <- read.table("Samburu_noNA.frq", h = T)
Sibiloi <- read.table("Sibiloi_noNA.frq", h = T)
Tsavo <- read.table("Tsavo_noNA.frq", h = T)

# Function for estimating the expected heterozygosity
het<-function(x){2*x*(1-x)}

# Remove all fixed alleles in each population
Amboseli <- Amboseli[Amboseli[, "MAF"] > 0, ]
Aruba_Dam <- Aruba_Dam[Aruba_Dam[, "MAF"] > 0, ]
Ikiri_Rungwa <- Ikiri_Rungwa[Ikiri_Rungwa[, "MAF"] > 0, ]
Masai_Mara <- Masai_Mara[Masai_Mara[, "MAF"] > 0, ]
Maswa <- Maswa[Maswa[, "MAF"] > 0, ]
Mkomazi <- Mkomazi[Mkomazi[, "MAF"] > 0, ]
Monduli <- Monduli[Monduli[, "MAF"] > 0, ]
Nairobi <- Nairobi[Nairobi[, "MAF"] > 0, ]
Samburu <- Samburu[Samburu[, "MAF"] > 0, ]
Sibiloi <- Sibiloi[Sibiloi[, "MAF"] > 0, ]
Tsavo <- Tsavo[Tsavo[, "MAF"] > 0, ]

total_sites <- 5909791 # from email by Genis

# and the pi-values for each polymorphic SNP
Amboseli <- cbind(Amboseli, pi=het(Amboseli$MAF) *(length(Amboseli$MAF)/(total_sites)))
Aruba_Dam <- cbind(Aruba_Dam, pi=het(Aruba_Dam$MAF) *(length(Aruba_Dam$MAF)/
                                                         (total_sites)))
Ikiri_Rungwa <- cbind(Ikiri_Rungwa,
                      pi=het(Ikiri_Rungwa$MAF)  *(length(Ikiri_Rungwa$MAF)/
                                                    (total_sites)))
Masai_Mara <- cbind(Masai_Mara, pi=het(Masai_Mara$MAF) *(length(Masai_Mara$MAF)/
                                                           (total_sites)))
Maswa <- cbind(Maswa, pi=het(Maswa$MAF) *(length(Maswa$MAF)/(total_sites)))
Mkomazi <- cbind(Mkomazi, pi=het(Mkomazi$MAF) *(length(Mkomazi$MAF)/(total_sites)))
Monduli <- cbind(Monduli, pi=het(Monduli$MAF) *(length(Monduli$MAF)/(total_sites)))
Nairobi <- cbind(Nairobi, pi = het(Nairobi$MAF) *(length(Nairobi$MAF) / (total_sites)))
Samburu <- cbind(Samburu, pi=het(Samburu$MAF) *(length(Samburu$MAF)/(total_sites)))
Sibiloi <- cbind(Sibiloi, pi=het(Sibiloi$MAF) *(length(Sibiloi$MAF)/(total_sites)))
Tsavo <- cbind(Tsavo, pi=het(Tsavo$MAF) *(length(Tsavo$MAF)/(total_sites)))

png('nucleotide_diversity.png')
par(mfrow = c(1, 1), cex.axis=0.6)
val = c(
  mean(Amboseli$pi),
  mean(Monduli$pi),
  mean(Nairobi$pi),
  mean(Mkomazi$pi),
  mean(Masai_Mara$pi),
  mean(Maswa$pi),
  mean(Samburu$pi),
  mean(Sibiloi$pi),
  mean(Aruba_Dam$pi),
  mean(Tsavo$pi),
  mean(Ikiri_Rungwa$pi)
```

```
)
names = c(
    "Amboseli", "Monduli", "Nairobi", "Mkomazi", "Masai M.", "Maswa",
    "Samburu", "Sibiloi", "Aruba D.", "Tsavo", "Ikiri_R.")
barplot(
  val,
  ylim = c(0.000, 0.0005),
  ylab = "pi",
  xlab = "Population",
  main = "Genetic differences by populations (localities)",
  col = c(rep("limegreen",4),
       "red", "red",
       "yellow", "yellow",
       "deepskyblue", "deepskyblue",
       "pink"),
  names.arg = names
)
dev.off()

pi_array <- array(val, dimnames = list(names, c("Pi")),
              dim = c(11,1))
pi_array
```

5. TREEMIX

5.1 (Look genetic diversity step 1 to 3)

Generated a .fam file as family ID file for each species and location and calculate allele frequency

5.2 Converted our files into treemix.frq

```
import sys, os, gzip

if len(sys.argv) < 3:
    print "plink2treemix.py [gzipped input file] [gzipped output file]"
    print "ERROR: improper command line"
    exit(1)
infile = gzip.open(sys.argv[1])
outfile = gzip.open(sys.argv[2], "w")

pop2rs = dict()
rss = list()
rss2 = set()

line = infile.readline()
line = infile.readline()
while line:
    line = line.strip().split()
    rs = line[1]
    pop = line[2]
    mc = line[6]
    total = line[7]
```

```
    if rs not in rss2:
        rss.append(rs)
    rss2.add(rs)
    if pop2rs.has_key(pop)==0:
        pop2rs[pop] =  dict()
    if pop2rs[pop].has_key(rs)==0:
        pop2rs[pop][rs] = " ".join([mc, total])
    line = infile.readline()

pops = pop2rs.keys()
for pop in pops:
    print >> outfile, pop,
print >> outfile, ""

for rs in rss:
    for pop in pops:
        tmp = pop2rs[pop][rs].split()
        c1 = int(tmp[0])
        c2 = int(tmp[1])
        c3 = c2-c1
        print >> outfile, ",".join([str(c1), str(c3)]),
    print >> outfile, ""
```

5.3 Used the TreeMix software to build the maximum likelihood tree

```
# plink2treemix already on the server and copied in current directory
mkdir -p treemix
cd treemix

# Change grants.fam file such as FID = species+.famFID and without IID:
cut -f 2 grants_popInfo.txt > temp
paste temp grants.fam > grants.fam
awk 'NR==FNR {h[$2] = $3; next} {print $1,$2,$4,$5,$6,$7}' grants.fam grants.fam
> grants_temp
cat grants_temp | grants.fam

# Create plink file with allele frequencies of each SNP species-wise:
plink --bfile grants --freq --missing --family --out freq_stat
gzip freq_stat.frq.strat

# Build treemix with outgroup as root and 2 migration events
python plink2treemix.py freq_stat.frq.strat.gz treemix.frq.gz
treemix -i treemix.frq.gz -root thomsons -m 2 -o treemix_results

# Create treemix figure in pdf file
Rscript plot.treemix.R treemix_results
evince treemix_results.pdf
```