Untitled

17022472

2021/2/23

Question 1

```
# import dataset

x = read.table("PopGenAssignment_Q1_data.txt",as.is=TRUE)
all.dates = as.integer(x[,1])
genetic.data = matrix(as.matrix(x[,-1]),ncol=dim(x)[2]-1)
dates = unique(all.dates)
num.snps = dim(genetic.data)[2]

# check dataset
nrow(genetic.data)
ncol(genetic.data)
# rows represent haplotypes and columns snps
```

a)

```
# assuming the "1" allele is simply the first allele

# adapting code given in the spreadsheet:
# res.mat <- matrix(rep(NA, ncol(genetic.data) * length(dates)),
# nrow = ncol(genetic.data), ncol = length(dates))
res.mean <- rep(NA, length(dates))
res.median <- rep(NA, length(dates))
for (i in 1:length(dates)){

  nhaps <- sum(all.dates == dates[i]) # no of haplotypes sampled at the ith date
  gen.dat <- genetic.data[all.dates == dates[i], ] # genetic data sampled at the
  # ith date
  allele.freq <- apply(gen.dat, 2, sum)/nhaps # apply() basically sums over column
  # gen.dat; since columns represent snps this will give each snps how many times
  # they were sampled at the ith date; divide by nhaps to give allele frequency
  # ( in population)
  # overall allele.freq is a 1000-length vector storing the allele freq of all snps
  # at time i. Now store all results in a table:
  res.mean[i] <- mean(allele.freq)
  res.median[i] <- median(allele.freq)
}

# the information on the first allele in the list, and some plots:

# l.allele.freq.change <- res.mat[1, ]
# plot(x = dates, y = l.allele.freq.change,
```

```
#      main = "Change in 1 allele frequency over time",
#      xlab = "Date of sampling",
#      ylab = "Allele frequency",
#      type = "l")

# and if what the question wants is the change of overall allele frequency?

par(mfrow = c(1,2))
plot(x = dates, y = res.mean,
     main = "Change in mean allele frequency over time",
     xlab = "Date of sampling",
     ylab = "Allele frequency",
     type = "l")

plot(x = dates, y = res.median,
     main = "Change in median allele frequency over time",
     xlab = "Date of sampling",
     ylab = "Allele frequency",
     type = "l")

par(mfrow = c(1,1))
# A joint plot
plot(x = dates, y = res.mean,
     main = "Change in mean and median allele frequency over time",
     xlab = "Date of sampling",
     ylab = "Allele frequency",
     type = "l",
     col = "red",
     ylim = c(0, max(res.mean) + 0.001))
lines(x = dates, y = res.median,
      col = "blue")
legend(x = 2000, y = 0.10,
       legend = c("Mean", "Median"),
       col = c("red", "blue"),
       lty = 1)
```

b)

```
res.prop <- rep(NA, length(dates))

for (i in 1:length(dates)){

  nhaps <- sum(all.dates == dates[i]) # no of haplotypes sampled at the ith
  # date
  gen.dat <- genetic.data[all.dates == dates[i], ] # genetic data sampled at
  # the ith date
  allele.freq <- apply(gen.dat, 2, sum)/nhaps # apply() basically sums over
  # column gen.dat; since columns represent snps this will give each snps how
  # many times they were sampled at the ith date; divide by nhaps to give allele
  # frequency ( in population)
  # overall allele.freq is a 1000-length vector storing the allele freq of all
  # snps at time i. Now store all results in a table:
  res.prop[i] <- sum(allele.freq==1)/num.snps # calculate the proportion of SNP
  # fixed with 1 at time i
```

```
}

plot(x = dates, y = res.prop,
     main = "Change in proportion of fixed SNPs over time",
     xlab = "date of sampling",
     ylab = "proportion",
     type = "l")
```

c)

```
wf <- function(npop=5,ngen=500,nall=100,init,mu=0)
  # this is a comment. Comments inform users; they are ignored by R
{
  freq = matrix(NA,npop,ngen) # create a matrix with npop rows and ngen columns,
  # with first columb being init
  freq[, 1] <- init
  for(i in 1:(ngen-1))
  {
    freq[,i+1] = rbinom(npop,nall,mu+(1-2*mu)*freq[,i]/nall) # create the 2,3,4,
    # ... ,ngen generations, in each of the npop populations, by random sampling
    # from the binomial distribution.  The expected allele fraction in the new
    # generation is the allele fraction in the current generation (freq[,i]/nall)
    # modified by the effect of mutation
  }
  # par(mar=c(4,4.5,2.5,0.5));matplot(t(freq),type="l",lty=1,lwd=2,xlab="generations",ylab="allele coun

  # just let it return with the entire freq matrix let us say
  return(freq)
}
```

```
data.time0 <- genetic.data[all.dates==dates[1],]
initial.allele.count <- apply(data.time0,2,sum)
```

```
nhap <- 80
wf.sim <- wf(npop = 1000, ngen = 10000, nall = nhap, init = initial.allele.count,
             mu = 0)
wf.sim.prop <- rep(NA, 10000)
for (i in 1:10000){
  wf.sim.prop[i] <- sum(wf.sim[, i] == nhap)/1000
}

plot(x = dates, y = res.prop,
     main = "Fixed allele proportion dynamics",
     xlab = "date of sampling",
     ylab = "proportion",
     type = "l",
     ylim = c(0, max(max(res.prop), max(wf.sim.prop))),
     col = "red")
lines(x = 1:10000, y = wf.sim.prop,
      col = "blue")
```

```
# To find which nhap "best resembles" the dataset I have to cycle through a range
# of possible values of nhaps:
options(warn=-1)
nhap.test <- 25:80
```

```
test.p.l <- rep(NA, length(nhap.test))

for (i in 1:length(nhap.test)){
  wf.sim <- wf(npop = 1000, ngen = 10000, nall = nhap.test[i],
               init = initial.allele.count, mu = 0)
  wf.sim.prop <- rep(NA, 10000)
  for (j in 1:10000){
    wf.sim.prop[j] <- sum(wf.sim[, j] == nhap.test[i])/1000
  }

  wf.sim.prop.compare <- wf.sim.prop[dates]

  # use a wilcoxon's test
  test.res <- wilcox.test(wf.sim.prop.compare, res.prop[-1])
  test.p.l[i] <- test.res$p.value
}
options(warn = 0)
```

d)

```
x <- read.table("PopGenAssignment_Q1_data_selection.txt", as.is = TRUE)

all.dates = as.integer(x[,1])
selected.SNP = matrix(as.matrix(x[,-1]),ncol=dim(x)[2]-1)

dim(selected.SNP)
range(all.dates)
dates <- seq(from = 0, to = 10000, by = 1000) # reset the dates, in case
# if the variable went missing
```

```
# frequency of 1 allele at the first SNP and its change over time:
freq.vec <- rep(NA,length(dates))
for (i in 1:length(dates))
{
nhaps <- sum(all.dates==dates[i])
freq.vec[i] <- sum(selected.SNP[all.dates==dates[i],1]==1)/nhaps
}


plot(x = dates, y = freq.vec,
     type = "l",
     main = "Allele frequency of first selected SNP over time",
     xlab = "Generations",
     ylab = "Allele frequency",
     ylim = c(0, 1.2))
```

```
stab.selected.SNP <- selected.SNP[all.dates == dates[1], ]
act.AA.count <- 0
act.Aa.count <- 0
act.aa.count <- 0
for (k in 1:(nrow(stab.selected.SNP)/2)){
  ind <- stab.selected.SNP[2*k-1, 2] + stab.selected.SNP[2*k, 2]
  if (ind == 0){
    act.aa.count <- act.aa.count +1
  }else if (ind == 1){
```

```
    act.Aa.count <- act.Aa.count +1
  }else if (ind == 2){
    act.AA.count <- act.AA.count +1
  }
}
AA.init <- act.AA.count/(nrow(stab.selected.SNP)/2)
Aa.init <- act.Aa.count/(nrow(stab.selected.SNP)/2)
aa.init <- act.aa.count/(nrow(stab.selected.SNP)/2)

pmat <- matrix(NA, ncol = max(dates)+1, nrow = 8)
sl <- seq(from = 0.01, to = 0.08, by = 0.01)

for (j in 1:8){
  AA <- AA.init
  Aa <- Aa.init
  aa <- aa.init
  p <- c()
  s <- sl[j]
  for (i in 1:(max(dates))){
    p[i] <- AA[i] + Aa[i] * 0.5
    total <- (1+s)*p[i]^2 + 2*p[i]*(1-p[i]) + (1-p[i])^2
    AA[i+1] <- (1+s)*p[i]^2 /total
    Aa[i+1] <- 2*p[i]*(1-p[i]) /total
    aa[i+1] <- (1-p[i])^2 /total
  }

  p[10001] <- AA[10001] + Aa[10001]*0.5
  pmat[j, ] <- p
}


# the following plots will not be demonstrated for simplicity.

# plot(x = dates, y = freq.vec, col = "red", type = "l", ylim = c(0, 1),
#      main = "Allele frequency: real versus simulated populations",
#      xlab = "Generations",
#      ylab = "Allele frequency at first selected SNP")
# legend.word <- c("real population")
# for (i in 1:8){
#   lines(x = dates, y = pmat[i, dates+1], col = i)
#   legend.word[i-1] <- paste("s =", sl[i], sep = " ")
# }
#
# legend("bottomright", legend = legend.word, col = c("red", 1:8), lty = 1)

# find threshold value, that is
search.fix <- function(x){
  # search over allele frequency list x and find the first occurance of 1,
  # returning generation number
  for (i in 1:length(x)){
    if (abs(x[i] - 1) < 0.000001){
      cat("Fixation reached at generation", i-1, "\n", sep = " ")
      break
```

```
    }
  }
}

# assess for all values of s in sl:
for (i in 1:length(sl)){
  cat("Selection coefficient =", sl[i], "\n")
  search.fix(pmat[i, ])
}
```

e)

```
# frequency of 1 allele at the first SNP and its change over time:
freq.vec <- rep(NA,length(dates))
for (i in 1:length(dates))
{
nhaps <- sum(all.dates==dates[i])
freq.vec[i] <- sum(selected.SNP[all.dates==dates[i],2]==1)/nhaps
}


plot(x = dates, y = freq.vec,
     type = "l",
     main = "Allele frequency of second selected SNP over time",
     xlab = "Date of sampling",
     ylab = "frequency",
     ylim = c(0, 1.2))
```

```
# simulation:
mean.allele.freq <- mean(freq.vec[-1]) # mean allele frequency for most times
exp.AA <- mean.allele.freq^2
exp.aa <- (1-mean.allele.freq)^2
exp.Aa <- 2* (1-mean.allele.freq) * mean.allele.freq
act.AA.count <- 0
act.Aa.count <- 0
act.aa.count <- 0
stab.selected.SNP <- selected.SNP
for (k in 1:(nrow(stab.selected.SNP)/2)){
  ind <- stab.selected.SNP[2*k-1, 2] + stab.selected.SNP[2*k, 2]
  if (ind == 0){
    act.aa.count <- act.aa.count +1
  }else if (ind == 1){
    act.Aa.count <- act.Aa.count +1
  }else if (ind == 2){
    act.AA.count <- act.AA.count +1
  }
}
act.AA <- act.AA.count/(nrow(stab.selected.SNP)/2)
act.Aa <- act.Aa.count/(nrow(stab.selected.SNP)/2)
act.aa <- act.aa.count/(nrow(stab.selected.SNP)/2)
act.freq <- matrix(c(act.AA, act.Aa, act.aa), nrow = 3)
exp.freq <- matrix(c(exp.AA, exp.Aa, exp.aa), nrow = 3)
res.mat <- cbind(act.freq, exp.freq)
rownames(res.mat) <- c("AA", "Aa", "aa")
colnames(res.mat) <- c("Actual frequency", "Expected frequency")
```

```r
act.AA.list <- rep(NA, length(dates))
act.Aa.list <- rep(NA, length(dates))
act.aa.list <- rep(NA, length(dates))
for (j in 1:length(dates)){

  act.AA.count <- 0
  act.Aa.count <- 0
  act.aa.count <- 0
  stab.selected.SNP <- selected.SNP[all.dates == dates[j], ]
  for (k in 1:(nrow(stab.selected.SNP)/2)){
    ind <- stab.selected.SNP[2*k-1, 2] + stab.selected.SNP[2*k, 2]
    if (ind == 0){
      act.aa.count <- act.aa.count +1
    }else if (ind == 1){
      act.Aa.count <- act.Aa.count +1
    }else if (ind == 2){
      act.AA.count <- act.AA.count +1
    }
  }
  act.AA.list[j] <- act.AA.count/(nrow(stab.selected.SNP)/2)
  act.Aa.list[j] <- act.Aa.count/(nrow(stab.selected.SNP)/2)
  act.aa.list[j] <- act.aa.count/(nrow(stab.selected.SNP)/2)
}

plot(x = dates, y = act.AA.list,
     col = "orange", ylim = c(0, 1), type = "l",
     xlab = "Generations",
     ylab = "Genotype frequency",
     main = "Change in genotype frequency over time")
lines(dates, act.Aa.list, col = "red")
lines(dates, act.aa.list, col = "blue")
abline(h = exp.AA, col = "orange", lty = 3)
abline(h = exp.Aa, col = "red", lty = 3)
abline(h = exp.aa, col = "blue", lty = 3)
legend("topright", legend = c("AA", "Aa", "aa"), col = c("orange", "red", "blue"), lty = 1)
```

```r
# Assuming the fitness for AA, Aa and aa were 1, 1, and 1-s respectively (s > 0),
# and such force remained constant
pmat <- matrix(NA, ncol = max(dates)+1, nrow = 8)
sl <- seq(from = 0.0001, to = 0.0008, by = 0.0001)

for (j in 1:8){
  AA <- c(act.AA.list[1])
  Aa <- c(act.Aa.list[1])
  aa <- c(act.aa.list[1])
  p <- c()
  s <- sl[j]
  for (i in 1:(max(dates))){
    p[i] <- AA[i] + Aa[i] * 0.5
    total <- p[i]^2 + 2*p[i]*(1-p[i]) + (1-s)*(1-p[i])^2
    AA[i+1] <- p[i]^2 /total
    Aa[i+1] <- 2*p[i]*(1-p[i]) /total
    aa[i+1] <- (1-s)*(1-p[i])^2 /total
```

```
  }

  p[10001] <- AA[10001] + Aa[10001]*0.5
  pmat[j, ] <- p
}

# plot(x = dates, y = p[dates+1],
#      type = "l",
#      col = "red",
#      ylim = c(0, 1))
# lines(x = dates, y = freq.vec, col = "blue")

plot(x = dates, y = freq.vec, col = "red", type = "l", ylim = c(0, 1),
     main = "Allele frequency: real versus simulated populations",
     xlab = "Generations",
     ylab = "Allele frequency at second selected SNP")
legend.word <- c("real population")
for (i in 3:8){
  target.p <- pmat[i, ]
  lines(x = dates, y = pmat[i, dates+1], col = i)
  legend.word[i-1] <- paste("s =", sl[i], sep = " ")
}

legend("topleft", legend = legend.word, col = c("red", 3:8), lty = 1)
```

Question 2:

```
pop1 = t(read.table("PopGenAssignment_Q2_chr16_pop1.haps"))
pop2 = t(read.table("PopGenAssignment_Q2_chr16_pop2.haps"))
pop3 = t(read.table("PopGenAssignment_Q2_chr16_pop3.haps"))
pop4 = t(read.table("PopGenAssignment_Q2_chr16_pop4.haps"))
pop5 = t(read.table("PopGenAssignment_Q2_chr16_pop5.haps"))
pop6 = t(read.table("PopGenAssignment_Q2_chr16_pop6.haps"))
# positions = as.integer(read.table("PopGenAssignment_Q2_chr16_SNPs.txt",as.is=T)[,2])

# for unknown reasons I have to import positions manually into
# an object called PopGenAssignment_Q2_chr16_SNPs
positions <- PopGenAssignment_Q2_chr16_SNPs[, 2]

dim(pop1)
dim(pop2)
dim(pop3)
dim(pop4)
dim(pop5)
dim(pop6)
length(positions)

# length of positions is rather 44500, not 44639. Anyway it agreed with the pop dataset.
# Organise all pop dataset together:
pop.list <- list(pop1, pop2, pop3, pop4, pop5, pop6)
pop.name <- rep(NA, 6)
for (i in 1:6){
  pop.name[i] <- paste("Population", i, sep = " ")
```

```
}
```

a)

```
# to calculate heterozygosity: for every two consecutive rows, is heterozygous
# if sum == 1, and not otherwise

med.het.calc <- function(pop, pos) {
  total.pop <- nrow(pop)/2
  het.l.pop <- rep(NA, length(pos))

  for (j in 1:length(pos)){

    het.count <- 0

    for (k in 1:total.pop){
      if ((pop[2*k-1, j]+pop[2*k, j]) == 1){
        het.count <- het.count + 1
      }

    }

    het.l.pop[j] <- het.count/total.pop
  }

  return(median(het.l.pop))
}
```

```
med.het.pop.list <- rep(NA, 6)
for (i in 1:6){
  med.het.pop.list[i] <- med.het.calc(pop.list[[i]], positions)
}

for (i in 1:6){
  cat(pop.name[i], "heterozygosity:", med.het.pop.list[i], "\n", sep = " ")
}
# All six pops show some levels of low heterozygosities. This seems to be common
# for human population? Also this might indicate there is no significant disparity between the six samp
```

b)

```
# for "pair" purpose, readapt the fst.calc() from the lecture:
fst.calc=function(x,y)
{
    all=c(x,y)
    p=length(all[all==0])/length(all)
    pX=sum(x==0)/length(x)
    pY=sum(y==0)/length(y)
    cX=length(x)/length(all)
    cY=length(y)/length(all)
    Fst=(sum(c(cX,cY)*(c(pX,pY)-p)^2))/(p*(1-p))
    return(Fst)
}

fst.mat <- matrix(NA, nrow = 6, ncol = 6)
```

```
for (i in 1:6){
  for (j in 1:6){
    first.pop <- pop.list[[i]]
    second.pop <- pop.list[[j]]
    fst.list <- rep(NA, length(positions))

    for (k in 1:length(positions)){
      fst.list[k] <- fst.calc(first.pop[, k], second.pop[, k])
    }

    fst.mat[i, j] <- median(fst.list, na.rm = TRUE)
  }
}


rownames(fst.mat) = colnames(fst.mat) = pop.name
fst.mat
```

c)

```
# Mutation rate
# Let us say a SNP value of 1 indicates mutation, and 0 not
mut.calc <- function(pop) {
  return(sum(pop)/(nrow(pop)*ncol(pop)))
  cat(sum(pop)/(nrow(pop)*ncol(pop)))
}

mut.l <- rep(NA, 6)
for (i in 1:6){
  mut.l[i]<- mut.calc(pop.list[[i]])
}

for (i in 1:6){
  cat(pop.name[i], "mutation rate:", mut.l[i], "\n", sep = " ")
}
```

All six populations have similar mutation rates.

d)

```
d.prime.calc=function(x,y)
{
  D.00 <- length(x[x==0 & y==0])/length(x)-(length(x[x==0])/length(x))*
    (length(y[y==0])/length(y))
  D.minus <- min((length(x[x==1])/length(x))*(length(y[y==1])/length(y)),
                 (length(x[x==0])/length(x))*(length(y[y==0])/length(y)))
  D.plus <- min((length(x[x==1])/length(x))*(length(y[y==0])/length(y)),
                (length(x[x==0])/length(x))*(length(y[y==1])/length(y)))
  if (D.00>=0) D.prime <- D.00/D.plus
  if (D.00<0) D.prime <- D.00/D.minus
  return(abs(D.prime))
}
```

```
LD.measure <- function(pop, snp.number = 1000,  pos = positions){
  n <- snp.number
  positions <- pos
```

```
  distances = D.prime.pop = r2.pop = matrix(NA,nrow=n,ncol=n)

  suppressWarnings(
  for (i in 1:(n-1)){
    for(j in (i+1):n){
      D.prime.pop[i,j] <- d.prime.calc(pop[,i],pop[,j])
      distances[i,j] <- abs(positions[i]-positions[j])
      r2.pop[i, j] <- cor(pop[, i], pop[, j])^2
    }
  }
  )

  dist.bins <- seq(0,50000,by=1000)
  mean.D.prime.pop <- rep(NA,length(dist.bins)-1)
  mean.r.square.pop <- rep(NA, length(dist.bins)-1)

  for (i in 1:(length(dist.bins)-1)){
    mean.D.prime.pop[i] <- mean(D.prime.pop[distances>dist.bins[i] &
                                              distances<dist.bins[i+1]],
                                na.rm=TRUE)
    mean.r.square.pop[i] <- mean(r2.pop[distances>dist.bins[i] &
                                          distances<dist.bins[i+1]],
                                 na.rm = TRUE)
  }


  # plot(x = dist.bins[-1], y = mean.D.prime.pop,
  #      main = paste("Mean |D'| values versus distance bin size",
  # "for the first", n, "SNP", sep = " "),
  #      xlab = "Distance bin size",
  #      ylab = "Mean |D'|",
  #      type = "l",
  #      col = "red",
  #      ylim = c(0, 1))
  # lines(x = dist.bins[-1], y = mean.r.square.pop,
  #      type = "l",
  #      col = "blue")
  # legend("topright", legend = c("D'", "r^2"), col = c("red", "blue"), lty = 1)

  res.mat <- rbind(mean.D.prime.pop, mean.r.square.pop)
  colnames(res.mat) <- dist.bins[-1]
  rownames(res.mat) <- c("Mean D'", "Mean r^2")

  return(res.mat)
  # the return is a matrix of two lists, storing mean D' and mean r^2 respectively
  # column names represent the distance between SNP pairs.

}
```

```
# full result for all populations

res1 <- LD.measure(pop1, snp.number = 1000, pos = positions)

res2<- LD.measure(pop2)
```

```
res3<- LD.measure(pop3)
res4 <- LD.measure(pop4)
res5 <- LD.measure(pop5)
res6 <- LD.measure(pop6)

x <- seq(from = 1000, to = 50000, by = 1000)
par(mfrow = c(1,2))
plot(x, y = res1[1, ],
     main = "Mean |D'| versus chromosomal\ndistance between SNPs",
     xlab = "Chromosomal distance (bp)",
     ylab = "Mean |D'|",
     type = "l",
     col = 1,
     ylim = c(min(cbind(res1[1, ], res2[1, ], res3[1, ], res4[1, ],
                        res5[1, ], res6[1, ])),
              max(cbind(res1[1, ], res2[1, ], res3[1, ], res4[1, ],
                        res5[1, ], res6[1, ]))))
lines(x, res2[1, ], col = 2)
lines(x, res3[1, ], col = 3)
lines(x, res4[1, ], col = 4)
lines(x, res5[1, ], col = 5)
lines(x, res6[1, ], col = 6)
legend("topright", legend = pop.name, col = 1:6, lty = 1)
plot(x = seq(from = 1000, to = 50000, by = 1000), y = res1[2, ],
     main = "Mean r^2 versus chromosomal\ndistance between SNPs",
     xlab = "Chromosomal distance (bp)",
     ylab = "Mean r^2",
     type = "l",
     ylim = c(min(cbind(res1[2, ], res2[2, ], res3[2, ], res4[2, ],
                        res5[2, ], res6[2, ])),
              max(cbind(res1[2, ], res2[2, ], res3[2, ], res4[2, ],
                        res5[2, ], res6[2, ]))))
lines(x, res2[2, ], col = 2)
lines(x, res3[2, ], col = 3)
lines(x, res4[2, ], col = 4)
lines(x, res5[2, ], col = 5)
lines(x, res6[2, ], col = 6)
legend("topright", legend = pop.name, col = 1:6, lty = 1)
```

e)

```
num.snps <- length(positions)
ind1.pop6=pop6[1:2,]
snps.tokeep=(1:num.snps)[apply(ind1.pop6,2,sum)==1]

positions.new <- positions[snps.tokeep]
pop1.new <- pop1[,snps.tokeep]
pop2.new <- pop2[, snps.tokeep]
pop3.new <- pop3[, snps.tokeep]
pop4.new <- pop4[, snps.tokeep]
pop5.new <- pop5[, snps.tokeep]
pop6.new <- pop6[, snps.tokeep]
pop.new.list <- list(pop1.new, pop2.new, pop3.new, pop4.new,
                     pop5.new, pop6.new)
```

```
pop.new.name <- rep(NA, 6)
for (i in 1:6){
  pop.new.name[i] <- paste("New population", i, sep = " ")
}
```

Repeat a) to d):

```
# for a)
med.het.list.new <- rep(NA, 6)
for (i in 1:6){
  med.het.list.new[i] <- med.het.calc(pop.new.list[[i]], positions.new)
}
for (i in 1:6){
  cat(pop.new.name[i], "heterozygosity score:", med.het.list.new[i],
      "\n", sep = " ")
  }
```

```
# b)
fst.mat <- matrix(NA, nrow = 6, ncol = 6)
for (i in 1:6){
  for (j in 1:6){
    first.pop <- pop.new.list[[i]]
    second.pop <- pop.new.list[[j]]
    fst.list <- rep(NA, length(positions.new))

    for (k in 1:length(positions.new)){
      fst.list[k] <- fst.calc(first.pop[, k], second.pop[, k])
    }

    fst.mat[i, j] <- median(fst.list, na.rm = TRUE)
  }
}
rownames(fst.mat) = colnames(fst.mat) = pop.new.name
fst.mat
```

```
# c):
mut.l.new <- rep(NA, 6)
for (i in 1:6){
  mut.l.new[i]<- mut.calc(pop.new.list[[i]])
}

for (i in 1:6){
  cat(pop.new.name[i], "mutation rate:", mut.l.new[i], "\n",
      sep = " ")
}
```

```
# d):

res1.new <- LD.measure(pop1.new, pos = positions.new)
res2.new <- LD.measure(pop2.new, pos = positions.new)
res3.new <- LD.measure(pop3.new, pos = positions.new)
res4.new <- LD.measure(pop4.new, pos = positions.new)
res5.new <- LD.measure(pop5.new, pos = positions.new)
res6.new <- LD.measure(pop6.new, pos = positions.new)
```

```
par(mfrow = c(1,2))
plot(x = seq(from = 1000, to = 50000, by = 1000), y = res1.new[1, ],
     main = "Mean |D'| versus chromosomal distance\n between SNPs",
     xlab = "Chromosomal distance (bp)",
     ylab = "Mean |D'|",
     type = "l", col = 1,
     ylim = c(min(res1.new[1, ], res2.new[1, ], res3.new[1, ],
                  res4.new[1, ], res5.new[1, ], res6.new[1, ]),
              max(res1.new[1, ], res2.new[1, ], res3.new[1, ],
                  res4.new[1, ], res5.new[1, ], res6.new[1, ])))
lines(x, res2.new[1, ], col = 2)
lines(x, res3.new[1, ], col = 3)
lines(x, res4.new[1, ], col = 4)
lines(x, res5.new[1, ], col = 5)
lines(x, res6.new[1, ], col = 6)
legend("topright", legend = pop.new.name, col = 1:6, lty = 1)
plot(x = seq(from = 1000, to = 50000, by = 1000), y = res1.new[2, ],
     main = "Mean r^2 versus chromosomal distance\n between SNPs",
     xlab = "Chromosomal distance (bp)",
     ylab = "Mean r^2",
     type = "l",
     ylim = c(min(res1.new[2, ], res2.new[2, ], res3.new[2, ],
                  res4.new[2, ], res5.new[2, ], res6.new[2, ]),
              max(res1.new[2, ], res2.new[2, ], res3.new[2, ],
                  res4.new[2, ], res5.new[2, ], res6.new[2, ])))
lines(x, res2.new[2, ], col = 2)
lines(x, res3.new[2, ], col = 3)
lines(x, res4.new[2, ], col = 4)
lines(x, res5.new[2, ], col = 5)
lines(x, res6.new[2, ], col = 6)
legend("topright", legend = pop.new.name, col = 1:6, lty = 1)
```