Simulations of population structure using the coalescent with recombination

Timothée Flutre (INRAE) modified by Jacques David

12/02/2020 10:52:04

Contents

```
suppressPackageStartupMessages(library(parallel))
suppressPackageStartupMessages(library(adegenet))
suppressPackageStartupMessages(library(rutilstimflutre)) # https://github.com/timflutre/rutilstimflutre
```

1 Computation context

Set up the parallel computations: If your computer has several cores it will mobilise all cores but one. This permits to speed up your analysis.

```r
nb.cores <- max(1, detectCores() - 1)
cl <- makeCluster(spec=nb.cores, type="PSOCK")
RNGkind("L'Ecuyer-CMRG")
clusterSetRNGStream(cl=cl, iseed=1234)
clusterEvalQ(cl, library(adegenet))
```

```
## [[1]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[2]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[4]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[5]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[6]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[7]]
## [1] "adegenet"  "ade4"      "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
```

2 Aim of the scripts

This script allows to generate diploid genotypes according to different genetic parameters such as genetic effective size, mutation rate, recombination rate and migration rate. It is based on the coalescence process.

3 Simulation of genotypes without structure

3.1 Population genetics parameters

In this case, we simulate a single population.
The parameters are the genetic effective size *Ne*, the chromosome length in base pairs, the mutation rate per base μ , the recombination rate per base r. So in this situation the population mutation rate is $\theta = 4N_e$ţ, the population recombination rate is $4N_e r$. They apply on chromosomes for which the length in base pair is given. The number of chromosomes is realized by repeating the process with the *nb.reps* value. The sample sizes is determined by *nb.genos*, not to be confounded with *Ne*.

```r
set.seed(1234)
nb.genos <- 300

# effective size
Ne <- 10^4

# Chromosome length in base pairs
chrom.len <- 10^5

# mutation rate per base
mu <- 10^(-8)

# recombination rate per base ?
c.rec <- 10^(-8)

# in the function nb.reps is the number of chromosoms
genomes.nostruct <- simulCoalescent(nb.inds=nb.genos, nb.reps=10,
                                    pop.mut.rate=4 * Ne * mu * chrom.len,
                                    pop.recomb.rate=4 * Ne * c.rec * chrom.len,
                                    chrom.len=chrom.len,
                                    nb.pops=1,
                                    verbose=1)
```

```
## Loading required namespace: scrm

## simulate according to the SCRM ...
## scrm 600 10 -t 40 -r 40 1e+05 -SC abs -oSFS
## nb of SNPs: 3005
##  chr1  chr2  chr3  chr4  chr5  chr6  chr7  chr8  chr9 chr10
##   262   297   271   329   347   348   302   268   280   301
## make a data.frame with SNP coordinates ...
## randomize haplotypes to make diploid genotypes ...
## convert haplotypes into genotypes encoded as allele dose ...
## permute alleles in haplotypes ...
## permute alleles in genotypes ...
```

3.2 Data structure

The output is a complex objec containing the command cmd, the coordinates of snp (snp.coords), the haplotypes of the two individual strands (haplos) on each chromosome haplos$chr_i and the genotypes (genos). The table *genos* is frame as usual, genotypes x snp.

```r
dim(genomes.nostruct$genos)
```

```
## [1]  300 3005
```

```r
genomes.nostruct$genos[1:4, 1:10]
```

```
##        snp0001 snp0002 snp0003 snp0004 snp0005 snp0006 snp0007 snp0008 snp0009
## ind001       0       2       2       2       0       2       2       0       0
## ind002       1       1       2       2       0       2       2       0       0
## ind003       1       1       2       2       0       2       2       0       0
## ind004       1       1       2       2       0       2       2       0       0
##        snp0010
## ind001       0
## ind002       0
## ind003       0
## ind004       0
```

4 Interconnected network of populations

4.1 Migration rates

The Island model is implemended. A new parameter has to be indicated, the number of migrants, N_em. The number of populations has also to be indicated, *nb.pops*.

Parameters are chosen to give consistent results. With low migration, we simulate more chromosomes, and increase the chromosome size in order to have more chance to observe polymorphic sites. The number of populations is nb.pops.

The chromsome lengths vary since it is less likely to observe polymorphisms in populations with low migration rate.

Three situations are simulated : high, medium and low.

```
chrom.lens <- c("high"=10^3, "med"=10^5, "low"=10^5)
mig.rates <- c("high"=10^4, "med"=10,   "low"=0.5)
nb.chroms <- c("high"=4,    "med"=4,    "low"=4)

genomes.struct <- list()

for(i in seq_along(mig.rates)){
  set.seed(1234)
  genomes.struct[[names(mig.rates)[i]]] <- simulCoalescent(nb.inds=nb.genos, nb.reps=nb.chroms[i],
                                                           pop.mut.rate=4 * Ne * mu * chrom.lens[i],
                                                           pop.recomb.rate=4 * Ne * c.rec * chrom.lens[
                                                           chrom.len=chrom.lens[i],
                                                           nb.pops=3, mig.rate=mig.rates[i],
                                                           verbose=1)
}
```

```
## simulate according to the SCRM ...
## scrm 600 4 -t 0.4 -r 0.4 1000 -I 3 200 200 200 10000 -SC abs -oSFS
## nb of SNPs: 43
## chr1 chr2 chr3 chr4
##   12    9    6   16
## make a data.frame with SNP coordinates ...
## randomize haplotypes to make diploid genotypes ...
## convert haplotypes into genotypes encoded as allele dose ...
## permute alleles in haplotypes ...
## permute alleles in genotypes ...
## simulate according to the SCRM ...
## scrm 600 4 -t 40 -r 40 1e+05 -I 3 200 200 200 10 -SC abs -oSFS
## nb of SNPs: 3462
## chr1 chr2 chr3 chr4
##  972  825  898  767
## make a data.frame with SNP coordinates ...
## randomize haplotypes to make diploid genotypes ...
## convert haplotypes into genotypes encoded as allele dose ...
## permute alleles in haplotypes ...
## permute alleles in genotypes ...
## simulate according to the SCRM ...
## scrm 600 4 -t 40 -r 40 1e+05 -I 3 200 200 200 0.5 -SC abs -oSFS
## nb of SNPs: 4636
## chr1 chr2 chr3 chr4
## 1168 1245 1106 1117
## make a data.frame with SNP coordinates ...
## randomize haplotypes to make diploid genotypes ...
## convert haplotypes into genotypes encoded as allele dose ...
## permute alleles in haplotypes ...
## permute alleles in genotypes ...
```

4.2 Data structure

There is now three lists of 4 objects. Each list is for one of the simulations. The sapply functions give the number of polymorphic SNP in each simulation. Each list is organised as before (cmd, snp.coords, haplos and genos).

```
summary(genomes.struct)
```

```
##      Length Class  Mode
## high 4      -none- list
## med  4      -none- list
## low  4      -none- list
```

```
sapply(genomes.struct, function(x){dim(x$genos)})
```

```
##      high  med  low
## [1,]  300  300  300
## [2,]   43 3462 4636
```

4.3 Visualisation

4.3.1 Combine all data into a single data set

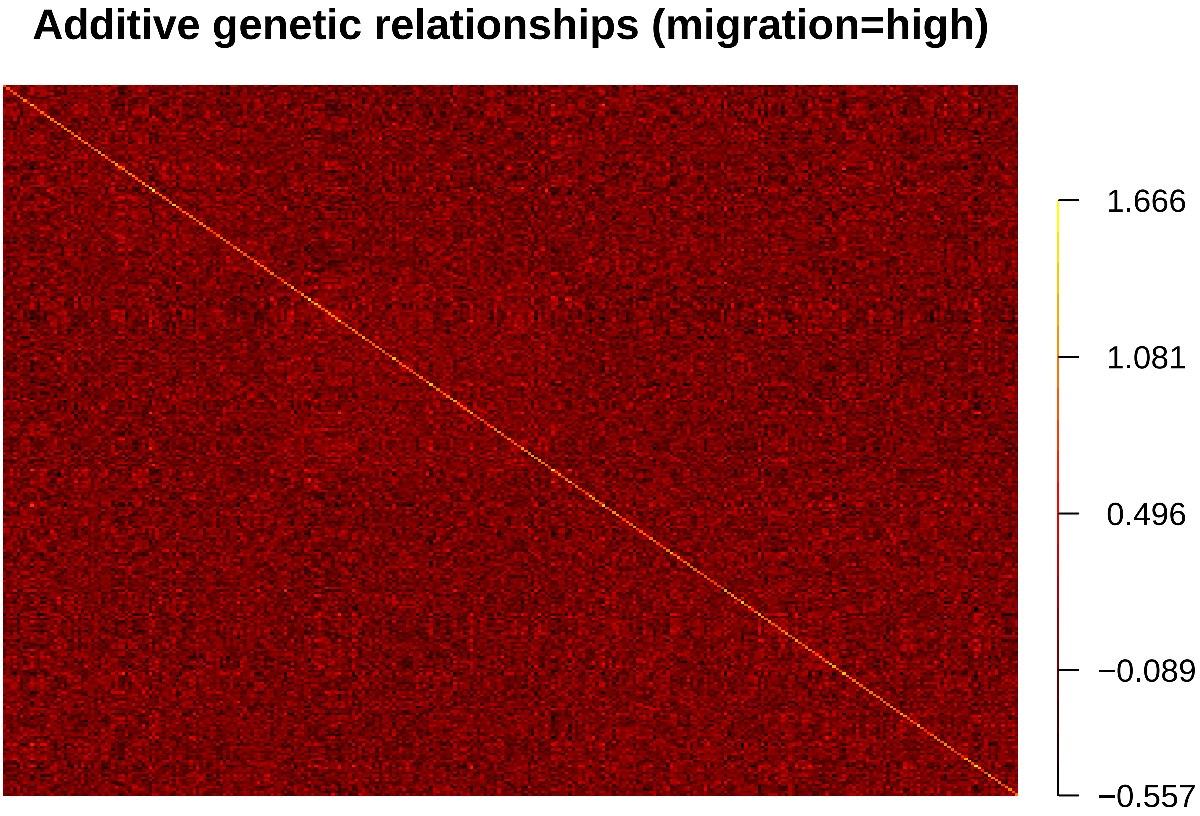
This is a complex programming but can help you to explore rapidly different situations in only one step. You can see in the Hands-on chapter to understand the basics.

```
X.pops <- lapply(names(mig.rates), function(n){
  tmp <- genomes.struct[[n]]$genos
  nb.remain.chrs <- 10 - length(unique(genomes.struct[[n]]$snp.coords[colnames(tmp),"chr"]))
  remain.chrs <- unique(genomes.nostruct$snp.coords[,"chr"])[1:nb.remain.chrs]
  snps.toadd <- rownames(genomes.nostruct$snp.coords[genomes.nostruct$snp.coords$chr %in%
                                                       remain.chrs,])
  X <- cbind(tmp, genomes.nostruct$genos[, snps.toadd])
  colnames(X)[(ncol(tmp)+1):ncol(X)] <- paste0(colnames(X)[(ncol(tmp)+1):ncol(X)],
                                               "_nostruct")
  X
})

names(X.pops) <- names(mig.rates)

sapply(X.pops, dim)
```

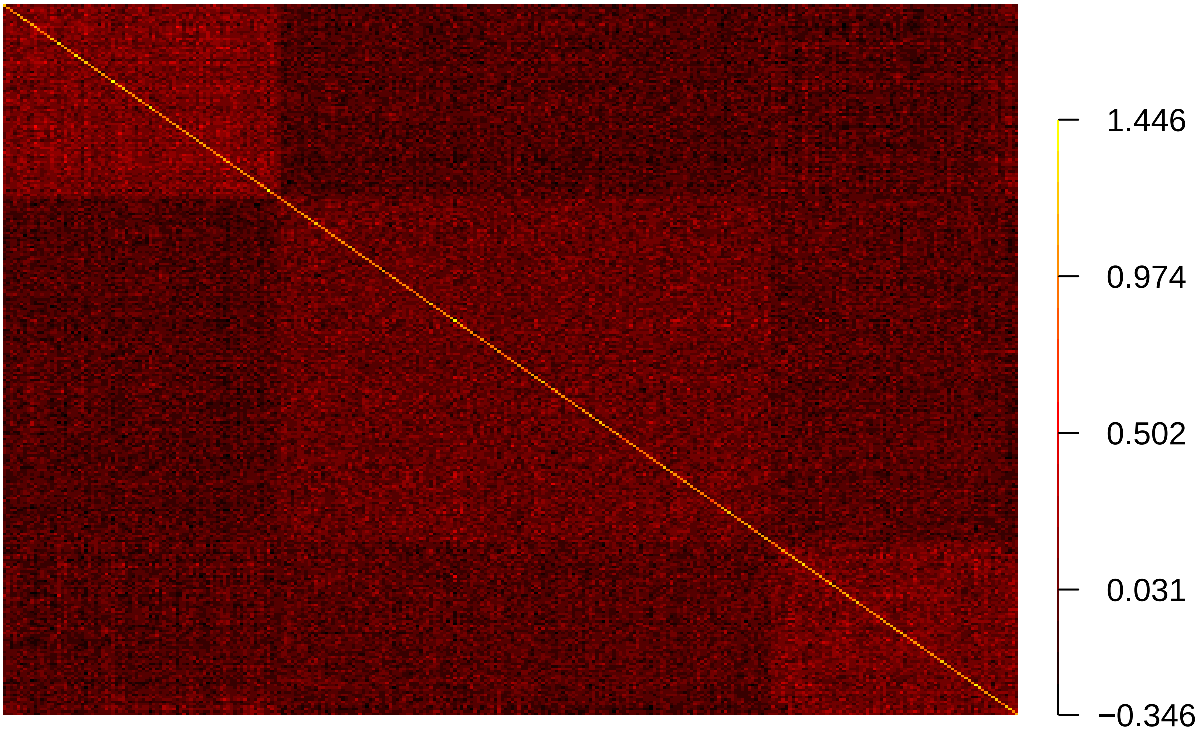
```
##      high  med  low
## [1,]  300  300  300
## [2,] 1897 5316 6490
```

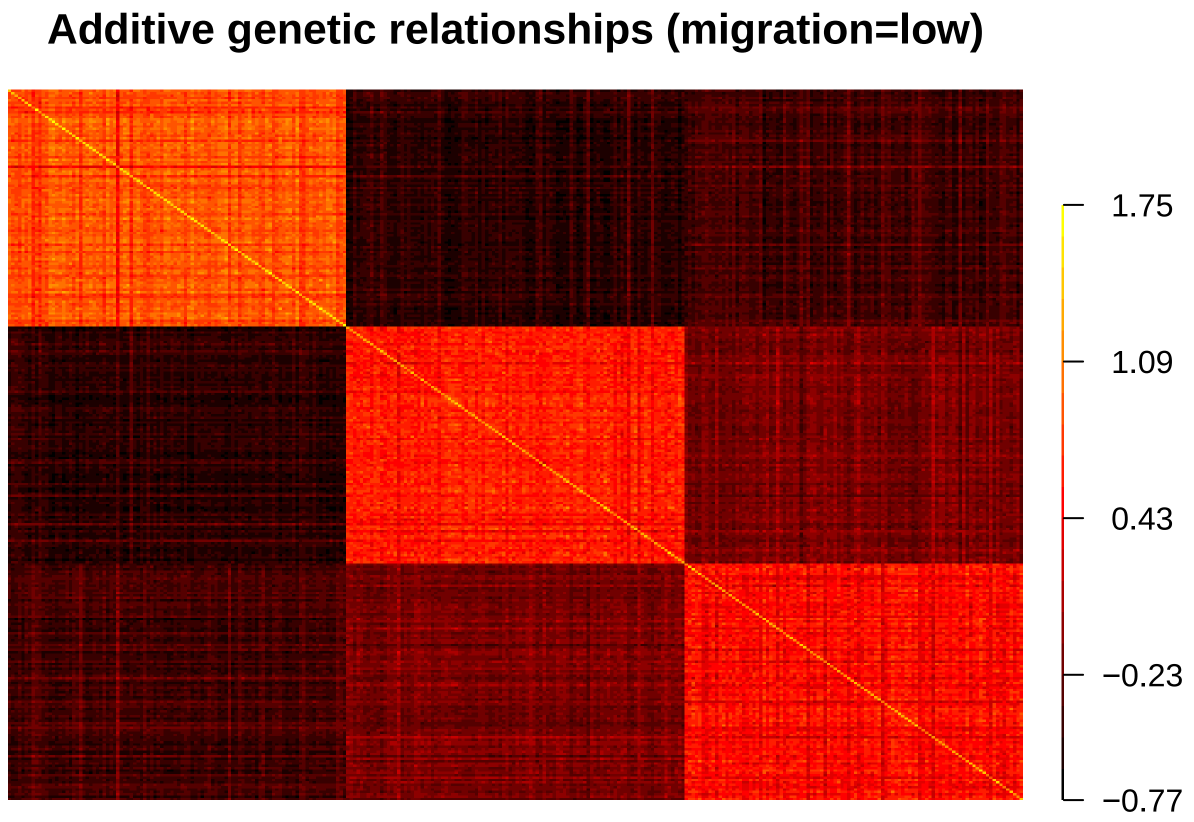
4.3.2 Relationship heatmap

```
tmp <- lapply(names(X.pops), function(n){
  A <- estimGenRel(X=X.pops[[n]], verbose=0)
  imageWithScale(A, main=paste0("Additive genetic relationships (migration=", n, ")"))
})
```



Additive genetic relationships (migration=med)



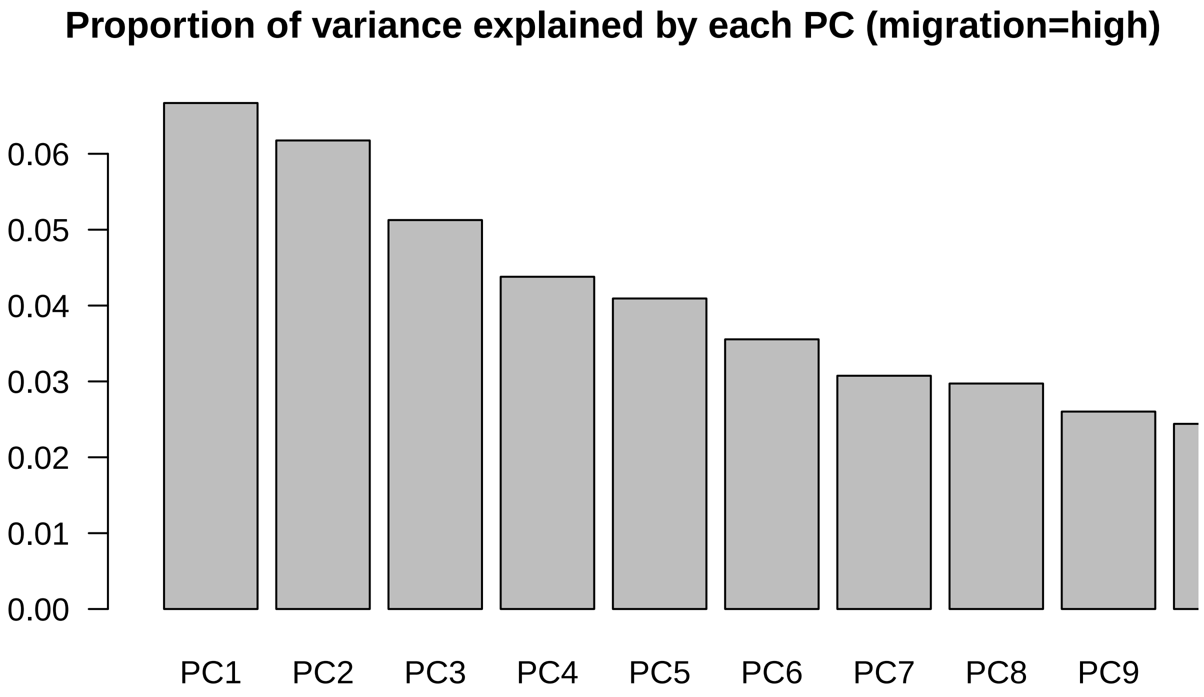


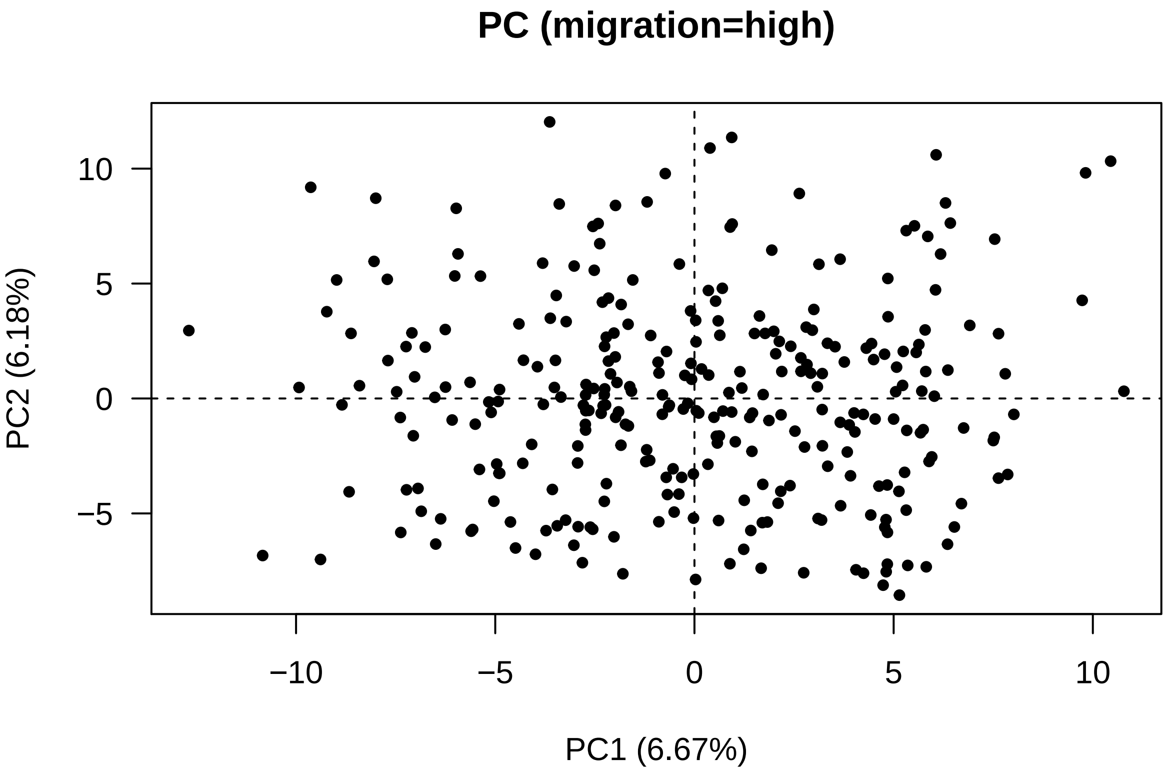
4.3.3 PcoA and clustering

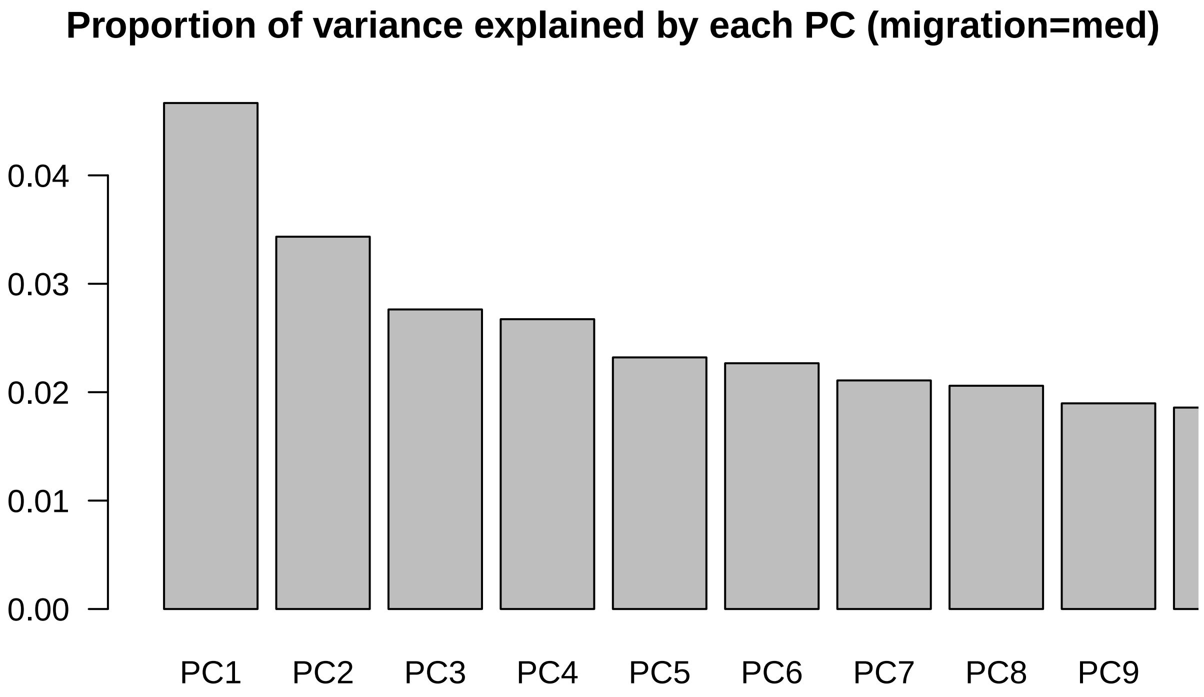
```
out.pca <- lapply(X.pops, function(X){
  pca(X=X)
})
sapply(out.pca, function(x){x$prop.vars[1:4]})
```

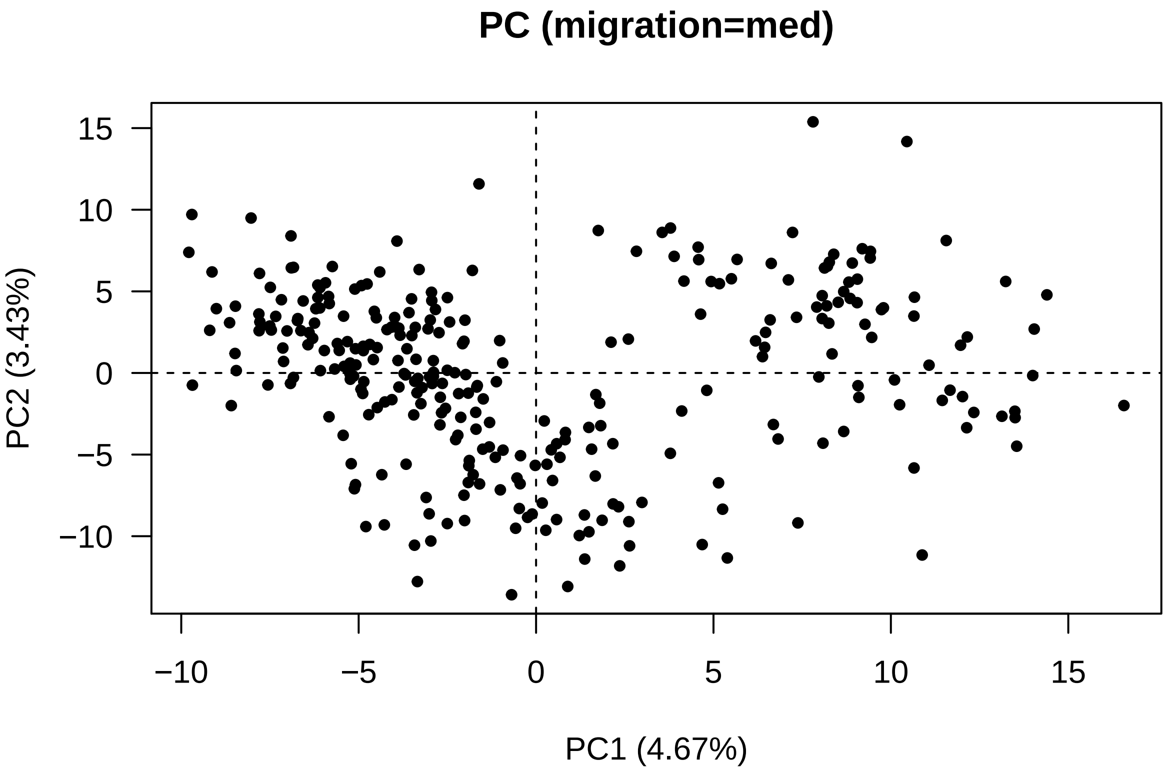
```
##           high        med        low
## PC1 0.06669451 0.04667145 0.33535779
## PC2 0.06175725 0.03433863 0.17893101
## PC3 0.05126737 0.02762894 0.01729453
## PC4 0.04379241 0.02673201 0.01529490
```

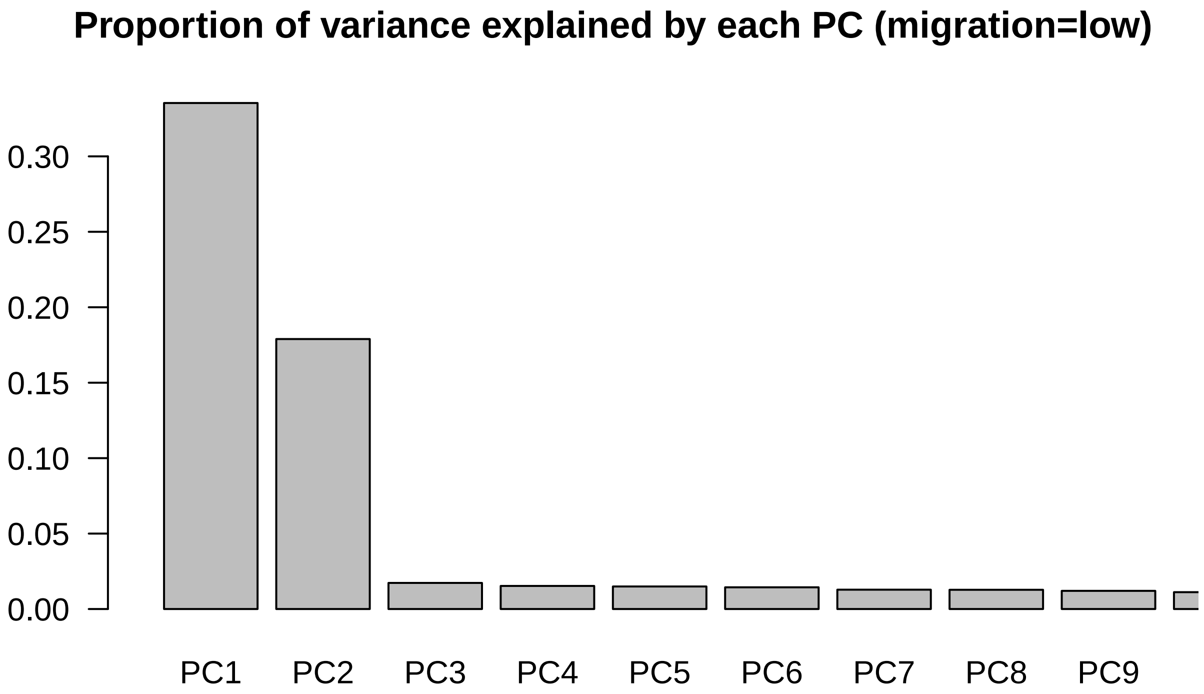
```
tmp <- lapply(names(out.pca), function(x){
  barplot(out.pca[[x]]$prop.vars,
          main=paste0("Proportion of variance explained by each PC (migration=", x, ")"),
          xlim=c(0,10), las=1)
  plotPca(rotation=out.pca[[x]]$rot.dat,
          prop.vars=out.pca[[x]]$prop.vars,
          # cols=c(rep("black", 100), rep("red", 100), rep("green", 100)),
          main=paste0("PC (migration=", x, ")"))
})
```

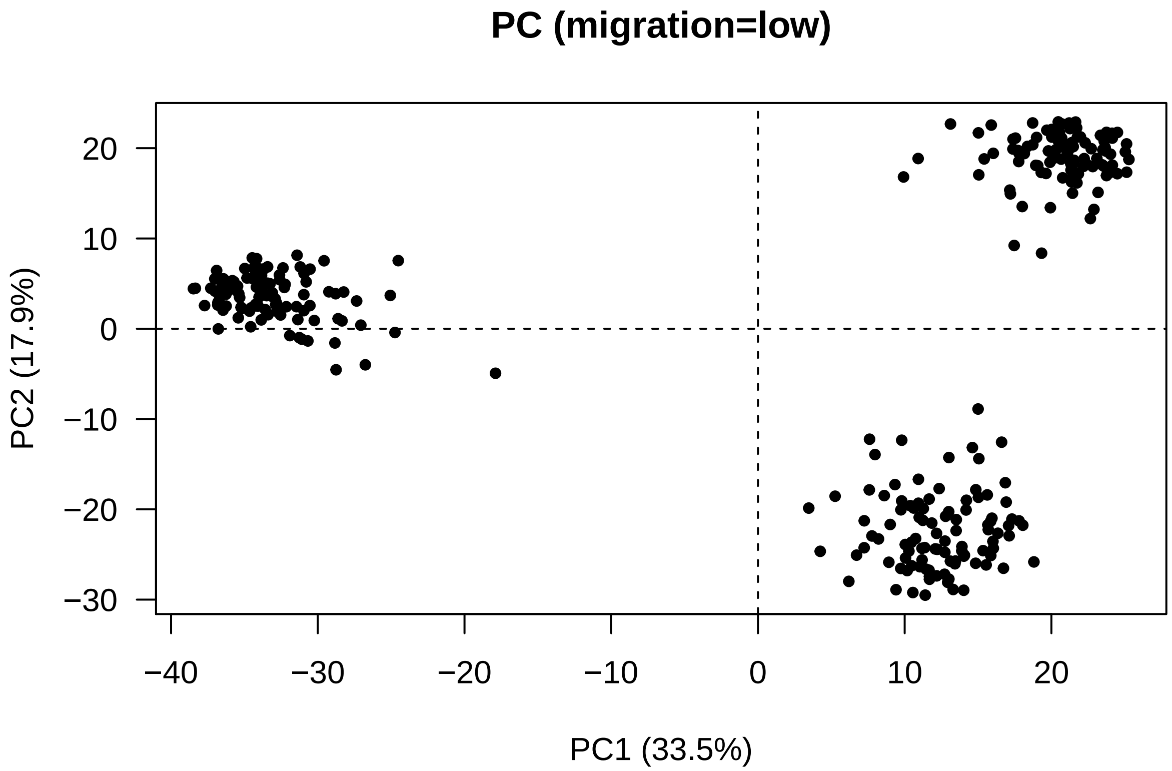

Proportion of variance explained by each PC (migration=high)
0.06
0.05
0.04
0.03
0.02
0.01
0.00
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9


PC (migration=high)
PC2 (6.18%)
10
5
0
−5
−10
−5
0
5
10
PC1 (6.67%)


Proportion of variance explained by each PC (migration=med)
0.04
0.03
0.02
0.01
0.00
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9


PC (migration=med)
PC2 (3.43%)
PC1 (4.67%)
15
10
5
0
−5
−10
−10
−5
0
5
10
15


Proportion of variance explained by each PC (migration=low)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PC8
PC9



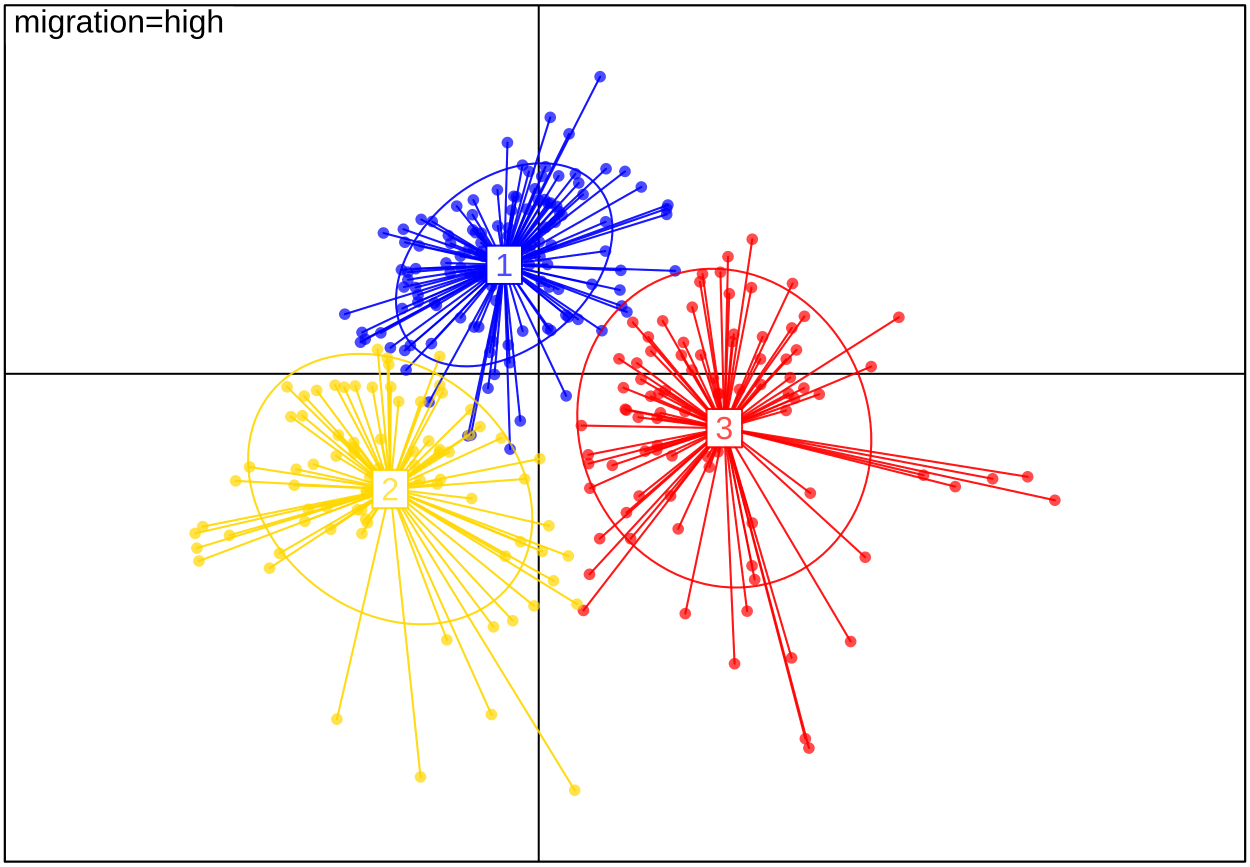
Need to get the point colors right! Let's use adegenet for this.

```
genlights <- lapply(X.pops, function(X){
  new("genlight", X)
})
fclusts <- parLapply(cl=cl, genlights, function(gl){
  find.clusters(x=gl, n.pca=100, scale=TRUE, method="kmeans",
                choose.n.clust=TRUE, n.clust=3)
                # stat="BIC", choose.n.clust=FALSE, max.n.clust=7, criterion="min")#smoothNgoesup")
})
sapply(fclusts, function(x){x$size})
```

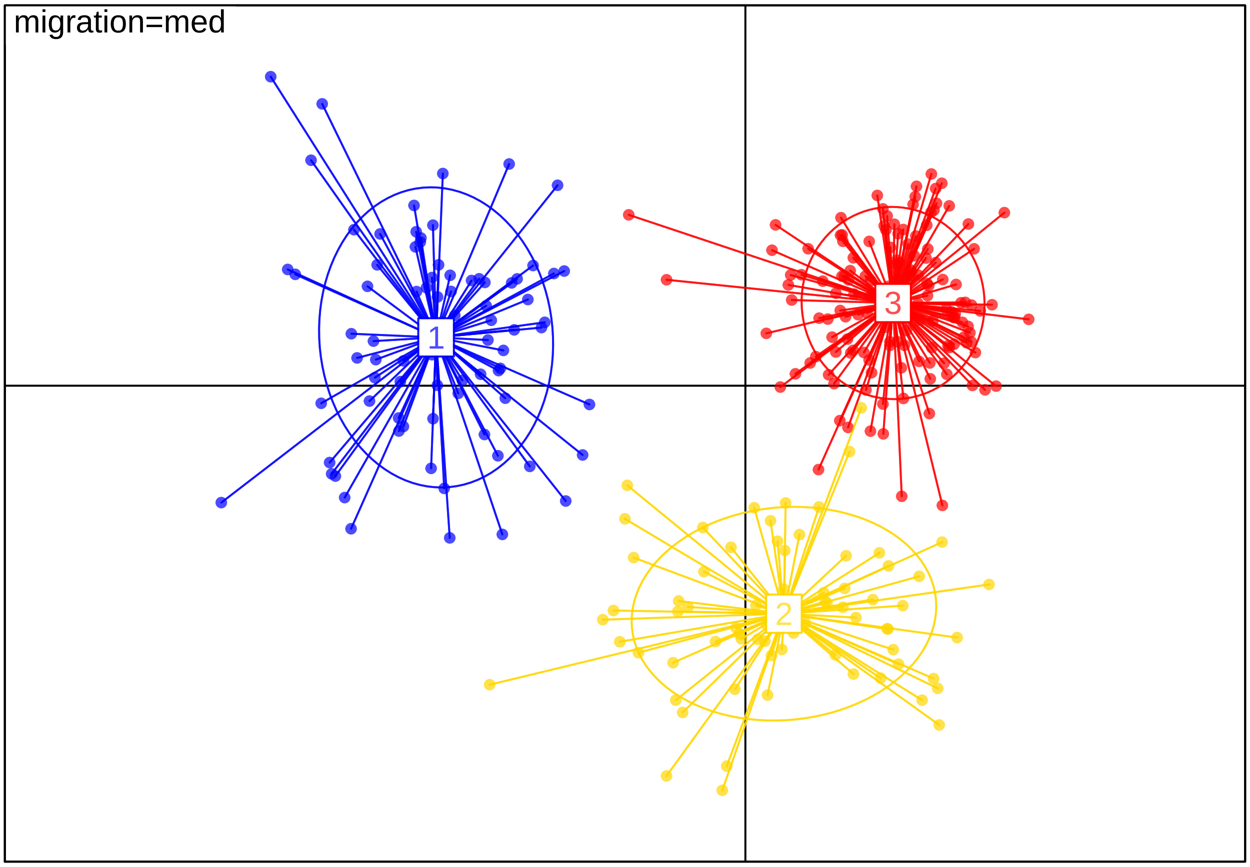
```
##      high med low
## [1,]  130  80 100
## [2,]   81  71 100
## [3,]   89 149 100
```

```
# tmp <- lapply(names(fclusts), function(x){
#   plot(fclusts[[x]]$Kstat, xlab="K", ylab="BIC",
#        main=paste0("Choose the number of clusters (migration=", x, ")"))
# })
clusterExport(cl=cl, varlist=c("genlights","fclusts"))
dapc <- parLapply(cl=cl, 1:length(genlights), function(i){
  dapc(x=genlights[[i]], pop=fclusts[[i]]$grp, n.pca=10, n.da=5)
})
```

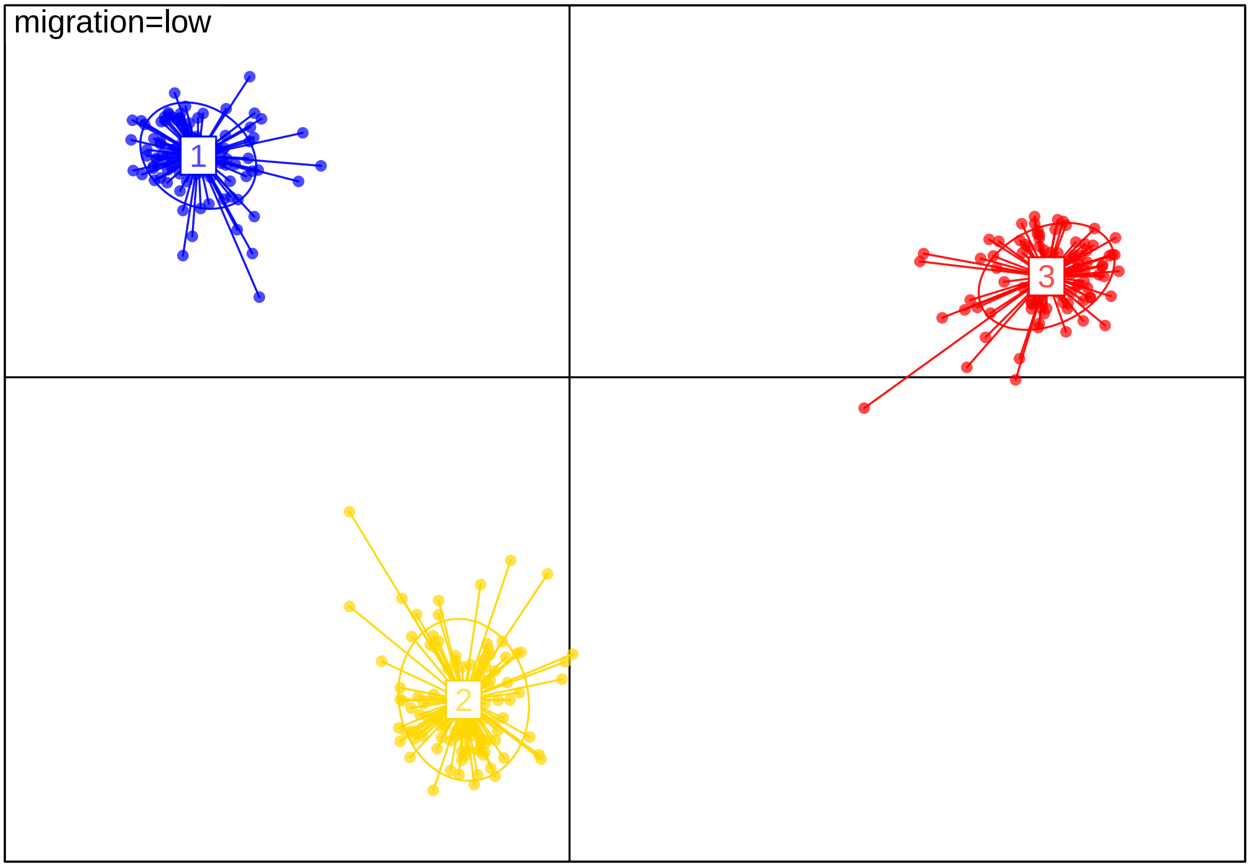
```r
names(dapc) <- names(genlights)
tmp <- lapply(names(dapc), function(x){
  print(scatter(x=dapc[[x]],
                sub=paste0("migration=", x), possub="topleft",
                scree.pca=FALSE, scree.da=FALSE))
})
```



```
## scatter.dapc(x = dapc[[x]], scree.da = FALSE, scree.pca = FALSE,
##     sub = paste0("migration=", x), possub = "topleft")
```



```
## scatter.dapc(x = dapc[[x]], scree.da = FALSE, scree.pca = FALSE,
##     sub = paste0("migration=", x), possub = "topleft")
```



```
## scatter.dapc(x = dapc[[x]], scree.da = FALSE, scree.pca = FALSE,
##     sub = paste0("migration=", x), possub = "topleft")
```

```
stopCluster(cl)
```

5 For your own Hands-on work

Here is a simplified version of the code. You can use it to explore your parameters. Be careful, this time *mig.rate* is the proportion of gametes coming from the pool of common gametes from all populations (migrants); So the parameter passed to the function is $4N_em$.

The expected overall Fst is $F_{st} = \frac{1}{1+(4N_em+\text{ţ})} \approx \frac{1}{1+4N_em}$. So the number of migrants has to be around 1 for the structuration to start to be high.

```
nb.chr<-4
nb.genos<-200

chrom.lens<-10^5
mig<-0.001
Ne<-5 * 10^3
mu<-1e-7
pop<-10

genomes.struct_case <- simulCoalescent(nb.inds=nb.genos, nb.reps=nb.chr,
             pop.mut.rate=4 * Ne * mu * chrom.lens,
             pop.recomb.rate=4 * Ne * c.rec * chrom.lens,
             chrom.len=chrom.lens,
             nb.pops=pop, mig.rate=4*Ne*mig,
             verbose=1)
```

```
## simulate according to the SCRM ...
## scrm 400 4 -t 200 -r 20 1e+05 -I 10 40 40 40 40 40 40 40 40 40 40 20 -SC abs -oSFS
## nb of SNPs: 57818
##  chr1  chr2  chr3  chr4
## 15293 14930 13378 14217
## make a data.frame with SNP coordinates ...
## randomize haplotypes to make diploid genotypes ...
## convert haplotypes into genotypes encoded as allele dose ...
## permute alleles in haplotypes ...
## permute alleles in genotypes ...
```

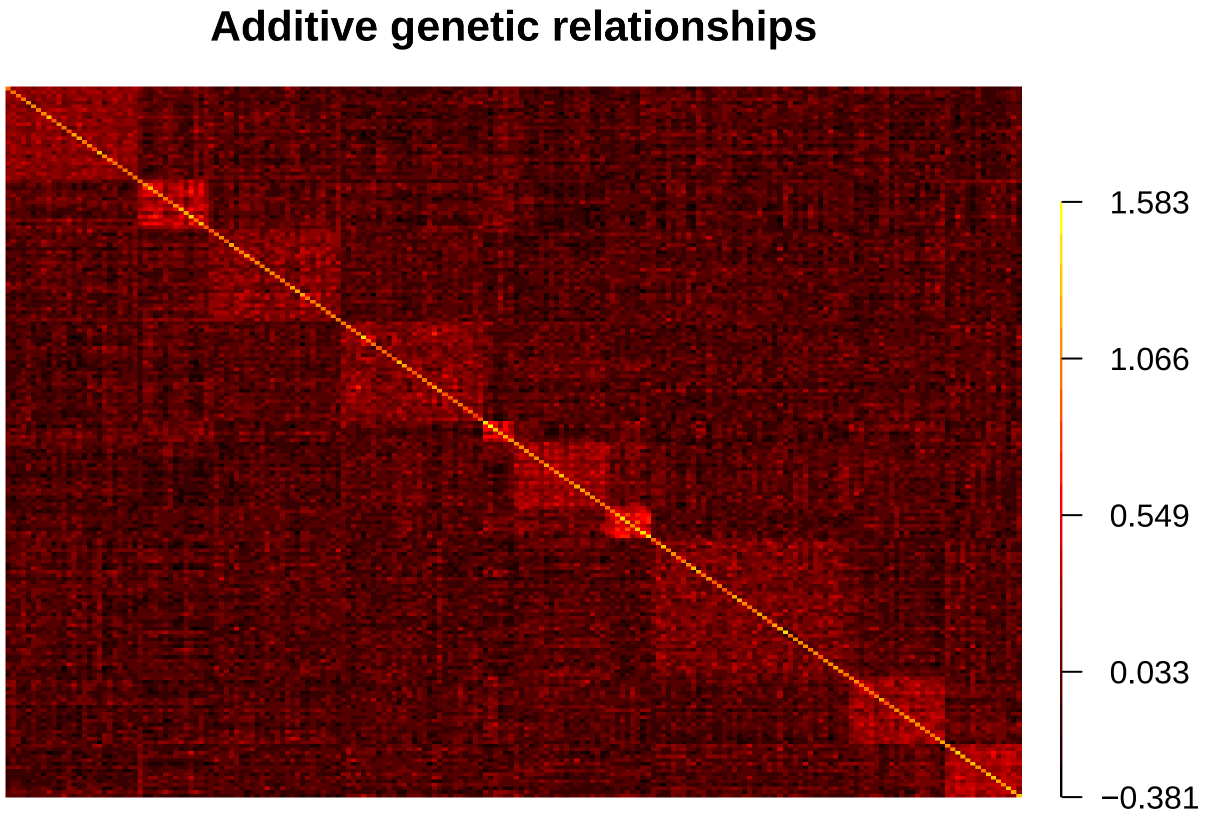
```
dim(genomes.struct_case$genos)
```

```
## [1]   200 57818
```

```
X<-genomes.struct_case$genos
A <- estimGenRel(X=X, verbose=0)
dim(A)
```

```
## [1] 200 200
```

```
imageWithScale(A, main="Additive genetic relationships")
```



The theoretical Fst value is 0.047619

Explore the parameters impact and use the data to explore the genomic prediction efficiency.