Source Code example for finding the correct balance for a European study and the exemplary target region chromosome 22

Required data:

Genotype data for chromosome 22 in .ped/.map format of external reference data, e.g. HapMap

Table with FID, IID, PID, MID, population, PC1, PC2, PC3 (where FID is the family ID, IID is the Individual ID, PID is the Paternal ID, MID is the Maternal ID)

snplist with SNPs available in your true study

recombination map for Hapmap (e.g. from impute website)

Step 1: find subgroups using R

```
#Input: forpca    Table with FID, IID, PID, MID, PC1, PC2, PC3
#                 for European study population
#       outfolder     directory for output

#Output: for each selection strategy one list of
#        individuals selected for sequencing and
#        one list of to-impute individuals is
#        saved in the output folder

#######################################################
#                                                     #
# 1 General Settings                                  #
#                                                     #
#######################################################

library(depth)   #for depth
library(reshape) #for sort_df
setwd(outfolder)

#######################################################
#                                                     #
# 2 Subset selection for sequencing                   #
#                                                     #
#######################################################
#univariate
#calculate univariate depth
for (i in 1:length(forpca[,1])){
forpca$tiefe[i]=depth(forpca[i,4],forpca[,4])*length(forpca[,4])}
#select 20 individuals with largest bivariate depth
subgroup=tail(sort_df(forpca, c("tiefe")), n=20)[,c(1:2,17)]
#save selection
write.csv(subgroup, file="Panelunivariate.csv")

#bivariate
#calculate bivariate depth
```

```
for (i in 1:length(forpca[,1])){
  forpca$tiefe[i]=depth(forpca[i,4:5],forpca[,4:5])* length(forpca[,4])}
#select 20 individuals with largest bivariate depth
subgroup=tail(sort_df(forpca, c("tiefe")), n=20)[,c(1:2,17)]
table(subgroup$POPULATI)
#save selection
write.csv(subgroup, file="Panelbivariate.csv")

#trivariate
#calculate trivariate depth
for (i in 1:length(forpca[,1])){
  forpca$tiefe[i]=depth(forpca[i,4:6],forpca[,4:6])*length(forpca[,4])}
#select 20 individuals with largest trivariate depth
subgroup=tail(sort_df(forpca, c("tiefe")), n=20)[,c(1:2,17)]
#save selection
write.csv(subgroup, file="Panel3D.csv")

#random1
set.seed(12345)
subgroup=forpca[sample(1:dim(forpca)[1], 20),]
write.csv(subgroup, file="Panelrandom.csv")

#random2
set.seed(3608)
subgroup=forpca[sample(1:dim(forpca)[1], 20),]
write.csv(subgroup, file="Panelrandom3608.csv")

#random3
set.seed(7616)
subgroup=forpca[sample(1:dim(forpca)[1], 20),]
write.csv(subgroup, file="Panelrandom7616.csv")

#random4
set.seed(4702)
subgroup=forpca[sample(1:dim(forpca)[1], 20),]
write.csv(subgroup, file="Panelrandom4702.csv")

#random5
set.seed(7784)
subgroup=forpca[sample(1:dim(forpca)[1], 20),]
write.csv(subgroup, file="Panelrandom7784.csv")

#######################################################
#                                                     #
# 2.3 list of to-impute                               #
#                                                     #
#######################################################
#none
excl_none=as.character(forpca$ID)
write.table(excl_none, file="excl_none.txt", quote=F, col.names=F,
row.names=F)
```

```
#random1
random=read.csv("Panelrandom.csv")
excl_random=as.character(forpca$ID[!forpca$ID %in% random$ID])
write.table(excl_random, file="excl_random.txt", quote=F, col.names=F,
row.names=F)

#random2
random=read.csv("Panelrandom3608.csv")
excl_random=as.character(forpca$ID[!forpca$ID %in% random$ID])
write.table(excl_random, file="excl_random3608.txt", quote=F,
col.names=F, row.names=F)

#random3
random=read.csv("Panelrandom7616.csv")
excl_random=as.character(forpca$ID[!forpca$ID %in% random$ID])
write.table(excl_random, file="excl_random7616.txt", quote=F,
col.names=F, row.names=F)

#random4
random=read.csv("Panelrandom4702.csv")
excl_random=as.character(forpca$ID[!forpca$ID %in% random$ID])
write.table(excl_random, file="excl_random4702.txt", quote=F,
col.names=F, row.names=F)

#random5
random=read.csv("Panelrandom7784.csv")
excl_random=as.character(forpca$ID[!forpca$ID %in% random$ID])
write.table(excl_random, file="excl_random7784.txt", quote=F,
col.names=F, row.names=F)

#univariate
univariate=read.csv("Panelunivariate.csv")
excl_univariate=as.character(forpca$ID[!forpca$ID %in% univariate$ID])
write.table(excl_univariate, file="excl_univariate.txt", quote=F,
col.names=F, row.names=F)

#bivariate
bivariate=read.csv("Panelbivariate.csv")
excl_bivariate=as.character(forpca$ID[!forpca$ID %in% bivariate$ID])
write.table(excl_bivariate, file="excl_bivariate.txt", quote=F,
col.names=F, row.names=F)

#3d
drei=read.csv("Panel3D.csv")
excl_3d=as.character(forpca$ID[!forpca$ID %in% drei$ID])
write.table(excl_3d, file="excl_3d.txt", quote=F, col.names=F,
row.names=F)
```

Step 2: subset to only European individuals and to study genotypes using plink

```
plink   --file data\affy6_chr22
        --keep out\EUR\EUR.keeplist
        --recode
        --out out\EUR_study
plink --file data\hm12_r27_chr22
      --keep out\EUR\EUR.keeplist
      --recode
      --out out\EUR_study_real_genotypes
```

Step 3: use gtool to convert study ped/map and true genotypes ped/map

```
./gtool/gtool -P --ped ./out/EUR/EUR_study.ped --map ./out/EUR/EUR_study.map --
og ./out/EUR/EUR.gen --os ./out/EUR/EUR.sampl
```

Step 4: Imputation using IMPUTE

```
./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_random3608.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_random3608.impute2 -allow_large_regions



./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_random7616.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_random7616.impute2 -allow_large_regions



./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_random4702.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_random4702.impute2 -allow_large_regions



./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
```

```
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_random7784.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_random7784.impute2 -allow_large_regions


./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_random.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_random.impute2 -allow_large_regions


./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref ./out/EUR/excl_3d.txt
-int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_3d.impute2 -allow_large_regions


./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_bivariate.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_bivariate.impute2 -allow_large_regions


./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_univariate.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_univariate.impute2 -allow_large_regions


./impute2 -seed 1164720 -m
./data/genetic_map_chr22_combined_b36.txt -g_ref ./out/hapmap.gen
-sample_g_ref ./out/hapmap.sampl -g ./out/EUR/EUR.gen -sample_g
./out/EUR/EUR.sampl -exclude_samples_g_ref
./out/EUR/excl_none.txt -int 14000000 50000000 -Ne 20000 -o
./out/EUR/1164720/Panel_none.impute2 -allow_large_regions
```

Step 5: Accuracy measures using R

```
##################################################################
# Input:    table1            IMPUTE2 output : table with        #
#                             imputed variants                   #
#           table2            table with all true variants       #
#           not_ incl         table with true variants that      #
#                             had been used                      #
#                             as 'typed ' in the simulation      #
#           n n               umber of to - impute individuals   #
# Output : cont_tables  List of contingency tables for all       #
#                             variants assumed 'untyped '        #
##################################################################

   cont_tables=function(table1, table2, not_incl ){
   tmp=table1[!(table1$V2 %in% not_incl$V2),] # exclude variants
considered as typed
   tmp_r=table2[(table2 $V2 %in% tmp$V2),]
   result = list()
   if(sum(as.character(tmp$V2)==as.character(tmp_r$V2))==dim(tmp)[1]){
                                          #check correct mapping
      for (k in 1: dim(tmp ) [1]) { # loop through all variants
         #initialize using first patient
         r=unlist(tmp_r[k, 6:8]) # three dosages ; real genotype
         i=unlist(tmp[k ,6:8]) # three dosages ; imputed genotype
         x=i%*%t(r)
         for (j in 2:n){ # loop through all remaining patients
            r= unlist (tmp_r[k, (3*j+3) :(3*j+5) ])
            i= unlist (tmp[k ,(3*j+3) :(3*j +5) ])
            x=x+(i%*%t(r))
         }
         result [[k]]=x
      }
   }
   return(result)
}

#######################################
# Input:    x     Contingency table   #
# Output:   Imputation Quality Score  #
#######################################

IQS=function(x){
   po=(x[1,1]+x[2,2]+x[3,3])/sum(x)

pc=(sum(x[,1])*sum(x[1,])+sum(x[,2])*sum(x[2,])+sum(x[,3])*sum(x[3,]))/(
sum(x)*sum(x))
   return((po-pc)/(1-pc))
}
```