Manual code: **`MSU_pigs_traits.R`**

Authors: *Jose Luis Gualdrón Duarte*[1] and *Juan Pedro Steibel*[2,3]

[1]Departamento de Producción Animal, Facultad de Agronomía, Universidad de Buenos Aires, Buenos Aires, ARG
[2]Department of Animal Science, Michigan State University, East Lansing, Michigan, USA
[3]Department of Fisheries and Wildlife, Michigan State University, East Lansing, Michigan, USA

Introduction

The R code `MSU_pigs_traits.R` is based on the paper Gualdrón et al. (2015). This code uses a gpData object called `pigMSU_traits (Load the object "pigMSU_B")` as input file, which contains different files: genotype file, map, pedigree, and phenotype file. For further details of construction and structure of gpData objects please refer to synbreed package in R (Wimmer et al. [1]).This package was used for assembling of gpData `pigMSU_traits`.

Description of Input Files

Each slot in `pigMSU_B` contains the following information:

Genotype file (pigMSU_B$geno)

In this file, animals IDs are row names and SNP names are column names. The genotypes are expressed as allelic dosage, having elements equal to 0,1, 2, i.e. the count of the allele used as reference, or a decimal number in the interval [0, 2] for imputed genotypes:

```
> pigMSU_B$geno[1:5,1:5]

      MARC0044150 ASGA0000014 ASGA0000021 ALGA0000009 ALGA0000014
1001            1           1           1           1           1
1002            1           1           1           1           1
1003            1           1           1           1           1
1004            1           1           1           1           1
1006            0           1           2           2           2
```

Map file (pigMSU_B$map)

This file contains SNP names as row names and also, chromosome and physical position expressed in Mega-bases as columns.

```
> head(pigMSU_B$map)

            chr      pos
MARC0044150   1 0.286933
ASGA0000014   1 0.342481
ASGA0000021   1 0.489855
ALGA0000009   1 0.538161
ALGA0000014   1 0.565627
H3GA0000032   1 0.573088
```

Pedigree file (pigMSU_B$pedigree)
The pedigree file contains five columns: animal ID - Sire ID - Dam ID - generation - sex (1=male, 2=female).

```
> head(pigMSU_B$pedigree)

    ID   Par1 Par2 gener sex
1 6070    0    0     0   0
2 6071    0    0     0   0
3 6086    0    0     0   0
4 6088    0    0     0   0
5 6092    0    0     0   0
6 6323    0    0     0   1
```

Phenotype file (pigMSU_B$pheno)
The trait file contains: animal ID in row names and the traits (28 traits in total) in column names:

```
> pigMSU_B$pheno[1:5,1:5,1]

      wt_birth wt_3wk wt_6wk wt_10wk bf10_10wk
1001      1.91   3.72  11.66   25.40      4.06
1002      1.91   8.21  19.55   35.38      9.40
1003      2.18   8.71  21.09   39.01      7.62
1004      2.00   8.57  19.87   38.10      6.86
1006      1.50   2.54   7.85   17.69      5.59
```

RUNNIG "`MSU_pigs_traits.R`":

Load required packages, functions and input files

First of all, code "`MSU_pigs_traits.R`" allows to load "Regress" package (version 1.3-10, R packages [2]) which is used in model fitting and variance components estimation process, performed by using REML algorithm.; and also, to load "Synbreed" package for data visualization and analysis of gpData `"pigMSU"`.

```
#################################################################
## R code MSU_pigs_traits.R                                    ##
## Paper: "Refining genome wide association for growth         ##
##         and fat deposition traits in a F2 pig population"  ##
#################################################################

#####################################
## 1) Load packages and read Rdata ##
#####################################

library(synbreed)
library(qvalue)
#unlock for install package qvalue
#source("http://bioconductor.org/biocLite.R")
#biocLite("qvalue")
library(snp.plotter)
source("Functions_codes_traits.R") # Load functions
```

Next, the gpData: "`pigMSU_traits`" and the R object: "`freq_geno_G_Z_pigMSU_traits.RData`" are uploaded:

```
#Load gpData
load("pigMSU_traits.Rdata") #Load the object "pigMSU_B"
#Load Znf2 and GTo matrix for F2 animals, also the F0 frequencies, F2 genotypes filtered by MAF
#(Note See line 260: "APPENDIX")
load("freq_geno_G_Z_pigMSU_traits.RData")
```

Fixed effects matrix construction

From object `"pigMSU_traits"` are extracted one by one the growth and fat deposition traits (28 traits in total, as example is used the growth trait: *16 week tenth rib backfat* (mm) or "bf10_16wk") and 2) sex for animals in the F_2, to be used as response variable ($\boldsymbol{y}$) and incidence matrix fixed effects ($\boldsymbol{X}$) for the model respectively.

```
########################################
## 2) Input files for funtion "gblup" ##
########################################
nt<-ncol(pigMSU_B$pheno) #number of traits
trait<-pigMSU_B$pheno
names_trait<-colnames(trait)

#for(i in 1:nt){ #unlock for all traits

i<-13 # example trait: 10th-rib backfat week 16 "bf10_16wk"

  ##Phenotype
  pheno<-pigMSU_B$pheno[,i,]

  indx<-match(rownames(Znf2),names(pheno))
  pheno<-pheno[indx]

  ##sex
  indx<-match(rownames(Znf2),pigMSU_B$pedigree$ID)
  sex<-pigMSU_B$pedigree$sex[indx]
  #model.matrix to create matrix X
  sex<-as.factor(sex)
  x<-model.matrix( ~ sex -1,
                   contrasts.arg=list(sex=contrasts(sex, contrasts=F)))

  #sex ID
  psex<-cbind(pigMSU_B$pedigree$ID,pigMSU_B$pedigree$sex)  #Extract  ID  and  sex  for  all  animals
(F0-F1 and F2)
  idxsex<-psex[,1]%in%names(pheno)
  sexid<-as.numeric(pigMSU_B$pedigree$ID[idxsex])
  rownames(x)<-sexid
  colnames(x)<-c("female","male")
```

Estimation of variance components and breeding values

Once incidence matrix **X** is constructed, **`"gblup"`** function allows to fit the model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{a} + \boldsymbol{e}$. This function takes as input the phenotypes file, **X** matrix, and also **Z** and **G** matrices. Internally, variance components are estimated by REML using `regress` package (version 1.3-10 R [2]).

```
#############################################################################
## 3) Apply function "gblup"                                                ##
##    return:                                                              ##
##    g_hat=SNPe,a_hat,E_variance,A_variance,Heredability,#iteratons,Ginv ##
#############################################################################

  trout_gblup<-gblup(pheno,x,Znf2,GTo)
  #save(trout_gblup,file=paste("trout_gblup_",names_trait[i],sep=""))
```

As a result, the function "**`gblup`**" gives by trait:

`llik`:	Estimate of the LogLikelihood for the model
`a_hat`:	Genome breeding values (GEBVs).
`E_variance`:	Error variance (σ_e^2)
`A_variance`:	Additive variance (σ_A^2)
`Heritability`:	Heritability of the trait
`n_iter`:	Number of iterations to converge.
`Ginv`:	The inverse of ***G*** matrix

Here, **`a_hat`** contains the random breeding values, such that $\boldsymbol{a} \sim N(0,\ \boldsymbol{G}\ \sigma_A^2)$, and $\boldsymbol{e}$ is the random error vector such that $\boldsymbol{e} \sim N(0, \boldsymbol{I}\ \sigma_e^2)$, and $\boldsymbol{I}$ is the identity matrix.

Estimation of SNP effects

The marker effect, variance of the markers effects, and *p*-values are calculated for each marker or SNP by trait. For this purpose, the function "**`snpe_GWA`**" is applied using elements obtained in function "**`snpe`**".

```
################################################################
## 4) Apply function "snpe_GWA"                              ##
##    return:                                                ##
##    beta=uh (SNPe "g_hat"), snp_variance=vsnp,pvalues=pvalue ##
################################################################

  #gwa_trait<-snpe_GWA(trout_gblup,x,Znf2) #unlock for all traits
  #save(gwa_trait,file=paste("gwa_",names_trait[i],sep="")) #unlock for all traits
  load("gwa_bf10_16wk")
```

As a result, the function "**`snpe_GWA`**" gives:

`beta`:	Estimate of each marker (SNP) effect
`snp_variance`:	Variance of each marker (SNP) effect
`pvalues`:	*p*-value for each marker (SNP)

Histogram of p-values and Manhattan plot by trait

Here, the histogram of *p*-values and the Manhattan plot by each trait are displayed. However, for the Manhattan plot is necessary the absolute marker (SNP) position or "consecutive position". Then, the function "**`abmap`**" is applied.

```
##############################
## 5) Map Absolute Position ##
##############################

  map<-as.matrix(pigMSU_B$map)              # Read map
  map1<-rownames(map)%in%colnames(Znf2)
  map2<-map[map1,]                          # Final SNP 40569
  #### Final map #####
  mapmsu1<-abmap(map2)                      # Apply funtion abmap
  idxc<-2-map2[,1]%%2                       # Color index


  # histogram of p-values
  #png(file=paste("hist_",names_trait[i],".png",sep=""))
  hist(gwa_trait$pvalues)
  #dev.off()

  # Manhattan Plot
  threshold<-0.05/nrow(map2)
  #pdf(pdf,file=paste("Manhattan_",names_trait[i],".pdf",sep=""))
  plot(mapmsu1,-log(gwa_trait$pvalues,10),pch=16,col=ifelse(idxc==2,"red","blue"),abline(h=-
log(threshold,10),lwd=1.1,col="red"),xlab="Absolute position Mb", ylab="-Log10(p-value)")
  #dev.off() #option to save the Manhattan plot in format ".pdf" in the current directory


########################################################
## 6) Highest -Log10(p-value) for chromosome and trait ##
########################################################
# assoc: information by character for all SNP filtered
#        given the p-value and -log10pvalue
  logpv<--log(gwa_trait$pvalues,10)
  assoc<-as.data.frame(cbind(map2,mapmsu1,gwa_trait$pvalues,logpv))
  colnames(assoc)<-c("chr", "pos_Mb","Abspos_Mb","pvalue","logpv")

#Extract max -logpvalues per chromosome
  result <- vector("list",18)
  for(j in 1:18){
    pvch<-assoc[assoc$chr==j,]
    maxpv<-which.max(pvch$logpv)
    mpv<-pvch[maxpv,]
    result[[j]] <-mpv
  }
  # List "maxlogpv" with the highest -log(p-value) per chromosome

  maxlogpv<-do.call(rbind,result)
  #save(maxlogpv,file=paste("maxlogpv_",names_trait[i],sep=""))

#} #unlock for all traits
```

False Discovery Rate (FDR)

The *p*-values by trait "`gwa_trait`" were filtered by FDR < 0.05, and saved on `"snpFDR.RData"`.

```
#############################
## 7) False Discovery Rate ##
#############################
snpFDR<-SNPs_FDR(gwa_trait,map2,0.05)
#save(snpFDR,file="snpFDR.RData")
```

Then, from "`snpFDR.RData`" the lowest *p*-values by chromosome and trait were selected, and saved on `"snp_mpvalueFDR.RData"`.

```
##########################################################################
## 8) Select minimum p-value in each trait for the SNP selected by FDR ##
##########################################################################
snp_mpvalueFDR<-pvalue_FDR(snpFDR)
#save(snp_mpvalueFDR,file="snp_mpvalueFDR.RData")
```

Definition of candidate segments of 2 and 6 Mega-bases

Candidate segments are defined by taking SNPs within one Mb upstream and one Mb downstream of the SNP with smallest *p*-value in each chromosome (see list `"snp_mpvalueFDR.RData"`).

```
################################
## 9) Segment of 2 Mega-bases ##
################################
```

```
## Extract SNP names selected #
snpnames<-unique(snp_mpvalueFDR[,1])

###############################################
# Critical p-valor for the segment significance #
###############################################
# 2800 Mb (Genome pig length aprox.)/2 Mb = 1400 Mb
# Bonferroni Correction (BC): 0.05/1400 = 3.571429e-05

seg_snp<- vector("list",length(snpnames))

for(i in 1:length(snpnames)){
  snp_trait<-subset(snp_mpvalueFDR,snp_mpvalueFDR[,1]==snpnames[i])
  # Apply function "propor_seg":
  seg_snp2<-
propor_seg(trait,map2,geno_f2_2,all_frq_f0,GTo,Znf2,dis_snp=1,snp_trait,chr=NULL,nameplow=NULL,na
mephigh=NULL)
  seg_snp[[i]]<-seg_snp2
}

#Object "results_segments" stores outputs of 2 Mb segments from function "propor_seg" in 2 Mb
segments
results_segments<-do.call(rbind, seg_snp)
```

The function "`proporseg`" use the markers SNP into the segment to create the matrices $\boldsymbol{Z}_1$ and $\boldsymbol{G}_1$, whereas genomic relationship matrix $\boldsymbol{G}_2$ was built using all remaining SNPs. Next, the model equal to: $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{e}$ ("`model2`"), where $\boldsymbol{a}_1$ is the vector of random effects associated with those SNP located in the segment, such that $\boldsymbol{a}_1 : N\left(0,\ \boldsymbol{G}_1 \sigma^2_{\mathrm{A}_1}\right)$ and $\boldsymbol{a}_2$ is the vector of additive random effects associated with all SNPs except those involved with $\boldsymbol{a}_1$, such that $\boldsymbol{a}_2 : N\left(0,\ \boldsymbol{G}_2 \sigma^2_{\mathrm{A}_2}\right)$. The `model2` is compared with the reduce model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{a} + \boldsymbol{e}$ ("`model1`") [4, 5] , from results obtained across the `regress` R package [2]. Finally, the function "`proporseg`" list the results obtained by `regress` R package [2] for "`model1`" and "`model2`", as follow:

`Loglikem2:`	LogLikehood "`model2`"
`LogLikem1:`	LogLikehood "`model1`"
`LRT:`	Likehood Ratio Test for "`model1`" and "`model2`"
`LRTseg:`	*p*-value for Likehood Ratio Test for the segment
`varE1:`	Error variance (σ^2_e) of "`model1`"
`varA1:`	Additive variance (σ^2_{A}) of "`model1`"
`varE2:`	Error variance (σ^2_e) of "`model2`"
`VarA2:`	Additive variance (σ^2_{A}) of "`model2`"
`Varseg:`	Additive variance segment $(\sigma^2_{\mathrm{A}_1})$ of "`model2`"
`Varseg_pr:`	Proportion in % of the total variance explained by the segment.

Results for function "**`proporseg`**" are stored in the object "**`results_segments`**"

```
#################################
## 10) Segment of 6 Mega-bases ##
#################################

##NOTE: As a result, after running all growth and fat deposition traits,
#       a genomic region of 6.2 Mega-bases was selected for multiple
#       traits (10 traits) on chromosome 6. This region is located between
#       131.8855 Mb and 138.0844 Mb,as describe in the paper Gualdron et al. (2015).
#       To make it more practrical,the complete table filtering by FDR<0.05 and the
#       lowest p-value by chromosome and trait "snp_mpvalueFDR.RData" is loaded.


#################################################
# Critical p-valor for the segment significance #
#################################################
# 2800 Mb (Genome pig length aprox.)/6 Mb = 466 Mb
# Bonferroni Correction (BC): 0.05/466 = 0.0001073

load("snp_mpvalueFDR.RData")        #Load the complete list (already running the total traits)
snpnames6<-subset(snp_mpvalueFDR,snp_mpvalueFDR[,3] == '6')

seg6_snp<- vector("list",nrow(snpnames6))
#for(i in 1:nrow(snpnames6)){     #unlock for all segments on chromosome 6.
  i<-1
  snp_trait6<-t(as.matrix(snpnames6[i,]))
  # Apply function "propor_seg"
  seg_snp6<-
propor_seg(trait,map2,geno_f2_2,all_frq_f0,GTo,Znf2,dis_snp=2,snp_trait6,chr=6,nameplow
="M1GA0008917",namephigh="ALGA0104402")
  seg6_snp[[i]]<-seg_snp6
#}                                 #unlock for all segments on chromosome 6.

#Object "results_segments6" stores outputs of 6 Mb segments from function "propor_seg".
results_segment6<-do.call(rbind, seg6_snp)
```

Linkage Disequilibrium Plot (LD-Plot)

Next, a Linkage Disequilibrium Plot (LD-Plot) for the segment of 6 Mega-base in each trait is created. As an example, the trait `"bf10_16wk"` on chromosome 6 is plotted.

```
############################################################
## 11) PLOT LD for segments of 6 Mega-bases on chromosome 6 ##
############################################################

#Pedi: "ped": Family-ID-sireID-damID-sex-column9
fped<-cbind(rep(1,nrow(pigMSU_B$pedigree)),pigMSU_B$pedigree[,c(1:3,5)],rep(-
9,nrow(pigMSU_B$pedigree)))
colnames(fped)<-c("Family","ID","Sire","Dam","sex","pheno")

### Filtering chromosome 6 ###
# This R object iclude all traits filtering by FDR and Lowest p-value/chromosome/trait.
snp_chr<-subset(snp_mpvalueFDR,snp_mpvalueFDR[,3]=="6")
snp_chr<-unique(snp_chr[,c(1,3,4),drop=FALSE])


# Segment chromosome 6 #
# SNP reference to select chromosome for 10 traits
```

```
schr<-as.numeric(unique((snp_chr[,2])))

# Range of the segment
pos_snp<-sort(as.numeric(snp_chr[,3]))
phigh<-pos_snp[length(pos_snp)]+2      #136.084448+2= 138.0844
plow<-pos_snp[1]-2                     #133.88546-2=131.8855


### Extract SNP in this range
mapslx<-subset(map2,map2[,1]==schr ) # Select the chromose in the map
slxmap<-subset(mapslx,(mapslx[,2]<=phigh)&(mapslx[,2]>=plow))
dim(slxmap) # [1] 88  2 (88 Markers in the segment)
slxnames<-rownames(slxmap) #

#significance traits in chromosome 6 =>

#Unlock to plot the 10 traits
#names_trait6<-
c("bf10_22wk","lrf_22wk","bf10_10wk","lrf_10wk","lrf_13wk","bf10_19wk","bf10_13wk","bf10_16wk","l
rf_19wk","lrf_16wk")

names_trait6<-c("bf10_16wk") # example trait

for(i in 1:length(names_trait6)){
  # load "gwa_trait" [1] "beta" "snp_variance" "pvalues"
  load(paste("gwa_",names_trait6[i],sep=""))
  pvt<-gwa_trait$pvalues #read pvalues

  ## Apply function "plot_ld"=>
  LD_seg6<-plot_ld(slxmap,pvt,fped,geno_f2_2,names_trait6[i])

  ### write the input files

write.table(LD_seg6$snpfile,file=paste("snpfile_",names_trait6[i],".txt",sep=""),sep="\t",row.nam
es=FALSE,col.names=TRUE,quote=FALSE) #

write.table(LD_seg6$genofile,file=paste("genofile_",names_trait6[i],".dat",sep=""),sep="\t",row.n
ames=FALSE,col.names=FALSE,quote=FALSE)
  write.table(LD_seg6$config,file=paste("config_",names_trait6[i],".txt",    sep=""),    append=F,
quote=F, col.names=F, row.names=F)

  ### Run program => config.txt (The configuration file for running snp.plotter) => Made a pdf
file with the LD-PLOT
  snp.plotter(config.file=paste("config_",names_trait6[i],".txt",sep=""))

}
```

As a result, the function "**`snp.plotter`**" gives a ".pdf " file with the LD-plot for the segment.

APPENDIX CODE:

Filter process by Minor Allele Frequency (MAF)
Genotypes in the F_2 haves a second editing process considering MAF < 0.05. As a result, a filtered matrix "**geno_f2_2**" that contains information for 928 F_2 animals with 40569 SNPs is created.

```
              ################################
              ##----------APPENDIX----------##
              ################################
#Load gpData
load("pigMSU_traits.Rdata") #Load the object "pigMSU_B"

#======== 1) FIRST FILTER F0 ANIMALS ACCORDING THEIR OWN FREQUENCIES ============

# Identify animals from the F0 generation
# Once gpData has been loaded, F0 animals should be identified considering their id number
# (F0 animals have id number > 6000)

# Define object related to genotypes file
geno<-pigMSU_B$geno
# Create a vector with animal ids
IDrow2M<-as.numeric(rownames(geno))
# Filter from previous vector, those related to F0 animals
geno0<-(IDrow2M>6000)
# Extract from geno file, those rows related to F0 animals
geno_f0<-geno[geno0,]
dim(geno_f0)    #[1]    19 44813   #That is, 19 animals F0.

# Compute allele frequencies for F0 animals: In this case, obtain means per column,
# removing "na" and divide by 2 (two alleles)
all_frq<-colMeans(geno_f0,na.rm=T)/2
# Check length of vector of allele freq
length(all_frq)                                           #[1] 44813
#
head(all_frq)
#MARC0044150 ASGA0000014 ASGA0000021 ALGA0000009 ALGA0000014 H3GA0000032
#  0.4210526   0.7894737   0.5789474   0.6052632   0.6052632   0.6315789

# Create an indicator function which will be equal to 1 if allele freq of SNP is <=0.5 or
# 0 otherwise (>0.5)
cond_alfrq<-(all_frq<=0.5)*1
head(cond_alfrq)
#MARC0044150 ASGA0000014 ASGA0000021 ALGA0000009 ALGA0000014 H3GA0000032
#          1           0           0           0           0           0
# Now, considering previous indicator function, compute minor allele freq for each SNP
# If allele freq is <=0.5, maf will be equal to the allele freq; if allele freq is >0.5,
# maf will be equal to 1-allele freq:
maf<-(1-cond_alfrq)+(((2*cond_alfrq)-1)*all_frq)
head(maf)
#MARC0044150 ASGA0000014 ASGA0000021 ALGA0000009 ALGA0000014 H3GA0000032
#  0.4210526   0.2105263   0.4210526   0.3947368   0.3947368   0.3684211
summary(maf)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#0.02632 0.15790 0.26320 0.27180 0.39470 0.50000

# Establish reference value for MINOR ALLELE FREQ: In this case, the threshold is
# 0.05 in order to keep rare alleles
MAF_ref<-0.05
# Identify with an index, those SNP with a MAF higher than defined threshold
maf_f0<-(maf>MAF_ref)
head(maf_f0)
#MARC0044150 ASGA0000014 ASGA0000021 ALGA0000009 ALGA0000014 H3GA0000032
#       TRUE        TRUE        TRUE        TRUE        TRUE        TRUE
# Create a matrix with those SNP having "TRUE" (those with MAF > 0.05)
# Use the indes to filter and identify them
SNPmaf<-as.matrix(all_frq[maf_f0])
length(SNPmaf)                                               #[1] 42979
summary(SNPmaf)
# Min.   :0.05263    1st Qu.:0.28947    Median :0.50000
# Mean   :0.50869    3rd Qu.:0.73684    Max.   :0.94737


#======== 2) SECOND, FILTER F2 ANIMALS ACCORDING FREQUENCIES FROM F0 ==============

# Identify F2 animals according to its id number (greater than 1000 and less than
# 6000)
genf2<-(IDrow2M>1000)&(IDrow2M<6000)
length(genf2)                                                 #[1] 1002
# Extract from initial geno file those rows related to F2 animals
genof2<-geno[genf2,]
dim(genof2)                                                   #[1] 928 44813
# 928 f2 animals

# Filter genotypes from F2 considering allele freq from F0, and specifically, discard
# those SNP that were filtered out by MAF>0.05 in F0 animals
geno_f2<-genof2[,colnames(genof2)%in%rownames(SNPmaf)]
dim(geno_f2)                                                  #[1] 928 42979


#======== 3) THIRD, FILTER F2 ANIMALS ACCORDING THEIR OWN FREQUENCIES =============
#
# Now, filter F2 generation according to its own frequency
# Thus, compute allele freq in F2 (mean by column divided by 2)
# Steps are the same as those used for F0
all_frq_f2<-colMeans(geno_f2,na.rm=T)/2
length(all_frq_f2)                                            #[1] 42979
# Use the same indicator function as in F0 animals
cond_alfrq_f2<-(all_frq_f2<=0.5)*1
maf_f2<-(1-cond_alfrq_f2)+(((2*cond_alfrq_f2)-1)*all_frq_f2)
summary(maf_f2)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.0000  0.1653  0.2837  0.2756  0.3911  0.5000

## Identify with an index, those SNP with a MAF higher than defined threshold
maf2<-(maf_f2>MAF_ref)
SNPmaf_f2<-as.matrix(all_frq_f2[maf2])
length(SNPmaf_f2)                                             #[1] 40569
summary(SNPmaf_f2)
# Min.   :0.05003   1st Qu.:0.30413   Median :0.50925
# Mean   :0.50678   3rd Qu.:0.71175   Max.   :0.94989


#======== 4) FOURTH, FILTER F0 ANIMALS ACCORDING FREQUENCIES FROM F2 ==============
#
# In this case, filter the last geno file obtained for F0 animals (geno_fo)
# according to frequencies in F2
genof0<-geno_f0[,colnames(geno_f0)%in%rownames(SNPmaf_f2)]
dim(genof0)                                                   #[1] 19 40569
# Compute allele frequencies again for F0
all_frq_f0<-colMeans(genof0,na.rm=T)/2
# Compute expected frequency (2p) in hardy Weinberg equilibrium
eg_f0<-2*all_frq_f0
length(eg_f0)                                                 #[1] 40569
summary(eg_f0)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.1053  0.6316  1.0000  1.0140  1.4210  1.8950

# Compute expected frequency for heterocigotes(2p*(1-p))
```

```
sdg_f0<-eg_f0*(1-all_frq_f0)
length(sdg_f0)                                              #[1] 40569
summary(sdg_f0)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#0.09972 0.30060 0.41140 0.38350 0.47780 0.50000

#####================================================================
# Now that double-way filters have been done, Z matrix can be computed, which
# contains centered allele dosages in terms of B allele

#============================================ COMPUTE Z MATRIX ==================
# Using heterogeneous procedure WITHOUT USE FUNCTION "ZSTANDARD"
# First,filter those snp in SNPmaf_f2 that are present in eg_f0 and sdg_f0 (to avoid warnings
# related to dimension inconsistencies)

geno_f2_2<-geno_f2[,colnames(geno_f2)%in%names(sdg_f0)]
dim(geno_f2_2)                                              #[1] 928 40569

# Check that SNP names are in the same order
index1<-match(names(eg_f0),colnames(geno_f2_2))
# As a way to check results from construcion of Z matrix with Z standard function,
# process was done using sweep operator
#  Sweep  operator  returns  an  array  obtained  from  an  input  array  by  sweeping  out  a  summary
statistic,
# in this case, sustracting from columns of "geno_f2_2"
# the expected frq obtained considering F0. Then, sweep operator is used again, but to perform
# division by expected standard deviation of gen. frq
# This was done following VanRaden (2008)
Zn_nofunct<-as.matrix(sweep(geno_f2_2[,index1],2,FUN="-",STATS=eg_f0))
Zn_nofunct<-as.matrix(sweep(Zn_nofunct,2,FUN="/",STATS=sqrt(sdg_f0)))
Zn_nofunct<-Zn_nofunct/sqrt(ncol(Zn_nofunct))
dim(Zn_nofunct)                                                 #[1] 928 40569
```

Construction of Z and G matrices

Following the approach of VanRaden [3], ***Z*** matrix for F_2 animals (`Znf2`) is calculated applying the function **"zstandard"**. Then, matrix ***G*** (`GTo`) for the same animals is calculated using the ***Z*** matrix obtained previously (`Znf2`).

```
# Apply zstandard function under HETEROGENEOUS procedure
# Using ZSTANDARD function, obtain Z matrix for F2 animals, considering heterogeneous
# procedure
Znf2<-zstandard(geno_f2_2,alfreq=all_frq_f0,procedure="heterogeneous")
dim(Znf2)                                   #[1] 928 40569


# Compare resulting matrix from ZSTANDARD with the one obtained in line 165
sum(Znf2-Zn_nofunct)                            #[1] 0
#Results are the same, thus, ZSTANDARD function works properly.

#============================================ COMPUTE G MATRIX ==================
# Once that Z matrix has been obtained, G matrix can be obtained as ZZ' (VanRaden, 2008)
GTo<-Znf2%*%t(Znf2)

# Both matrices (Z and G) will be used to fit a GBLUP model for variance components
# estimation.Thus, save them together with the "all_frq_f0","geno_f2_2", as "RData file"
```

Finally, a R object `"freq_geno_G_Z_pigMSU_traits.RData"` is created, saving the R objects: F_0 frequencies (`all_frq_f0`), F_2 genotypes filtered by MAF and matrices `Znf2` y `GTo` for the F_2 animals.

```
# Save outputs==>
#save(all_frq_f0,geno_f2_2,Znf2,GTo,file="freq_geno_G_Z_pigMSU_traits.RData")
```

To cite this code use:

J. L. Gualdrón Duarte, R. J. C. Cantet, Y. L. Bernal Rubio, R. O. Bates, C. W. Ernst, N. E. Raney, A. Rogberg-Muñoz, J. P. Steibel. **Refining genome wide association for growth and fat deposition traits in a F2 pig population**. *Journal of Animal Science* 2015. (*Submitted*)

REFERENCES

1. Wimmer V, Albrecht T, Auinger H-J, Schön C-C: **synbreed: a framework for the analysis of genomic prediction data using R.** *Bioinformatics* 2012, **28**:2086–7.

2. Clifford D, McCullagh P: **The regress function**. *R News* 2006, **6**:6–10.

3. VanRaden PM: **Efficient methods to compute genomic predictions.** *J Dairy Sci* 2008, **91**:4414–23.

4. Gualdrón Duarte JL, Cantet RJ, Bates RO, Ernst CW, Raney NE, Steibel JP: **Rapid screening for phenotype-genotype associations by linear transformations of genomic evaluations**. *BMC Bioinformatics* 2014, **15**:246.

5. Hayes BJ, Pryce J, Chamberlain AJ, Bowman PJ, Goddard ME: **Genetic architecture of complex traits and accuracy of genomic prediction: coat colour, milk-fat percentage, and type in Holstein cattle as contrasting model traits.** *PLoS Genet* 2010, **6**:e1001139.