First Estimates

Lucas Ramalho Anderson

21/10/2020

```
library ( dplyr )
# library ( plyr )
library ( ggplot2 )

tempDir = "/scratch/genevol/users/lucas/"
saveDir = "/raid/genevol/users/lucas/heritability/plots/"
```

Introduction

Step 1

After the removal of monomorphisms, filtration of the desired samples and after obtaining the list of non-correlated snp's per chromosome (considering correlation value of $\sqrt{0.1}$), it is now desired to calculate the GRM matrix.

```
# Read file with all chromosomes
# allChrFile = SeqArray::seqOpen ( paste0 ( tempDir , "allChr.gds" ) )
# List of all genes of interest (after pruning)
listSnps = readRDS ( paste0 ( tempDir , "fullPrunedList.rds" ) )

# GRM - calculated as defined in CGTA
# grm_obj = SNPRelate::snpgdsGRM( allChrFile , snp.id = listGenes , method = "GCTA")

# Estimating through "gaston" package
altReadSnps = gaston::read.vcf( paste0 ( tempDir , "allChr.vcf.gz" ) )
```

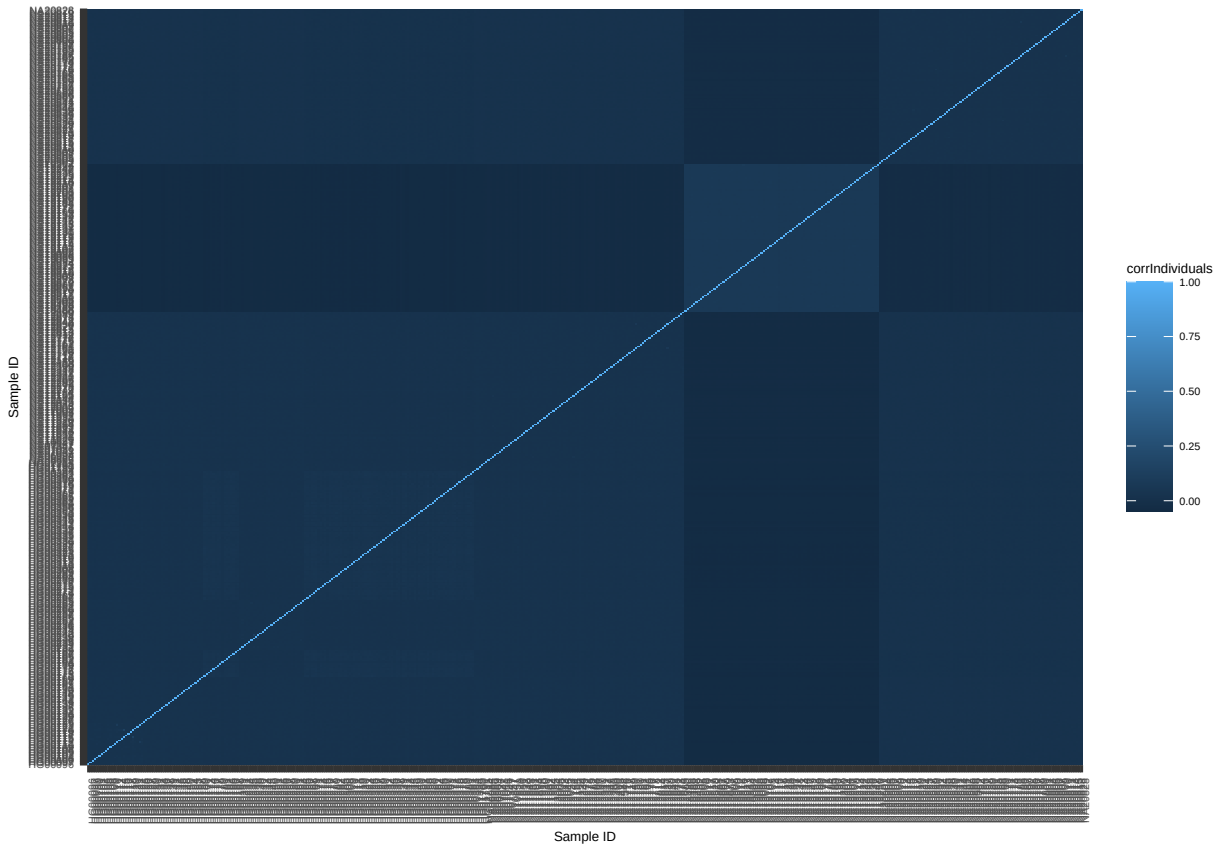
```
## ped stats and snps stats have been set.
## 'p' has been set.
## 'mu' and 'sigma' have been set.
```

```
# setting "p" parameter - correction with mean "p" and std sqrt(2p(1-2p))
gaston::standardize( altReadSnps ) <- "p"
grm_matrix = gaston::as.matrix ( altReadSnps )
# grm_scaled = scale( grm_matrix , center = T , scale = T )
# grm_scaled = readRDS (paste0(tempDir , "scaledMatrixBk.rds"))


# manual_GRM = ( 1 / nrow ( grm_scaled ) ) * grm_scaled %*% t ( grm_scaled )
# GRM matrix calculation (GCTA)
grm_alt_p = gaston::GRM ( altReadSnps , which.snps = listSnps )
```

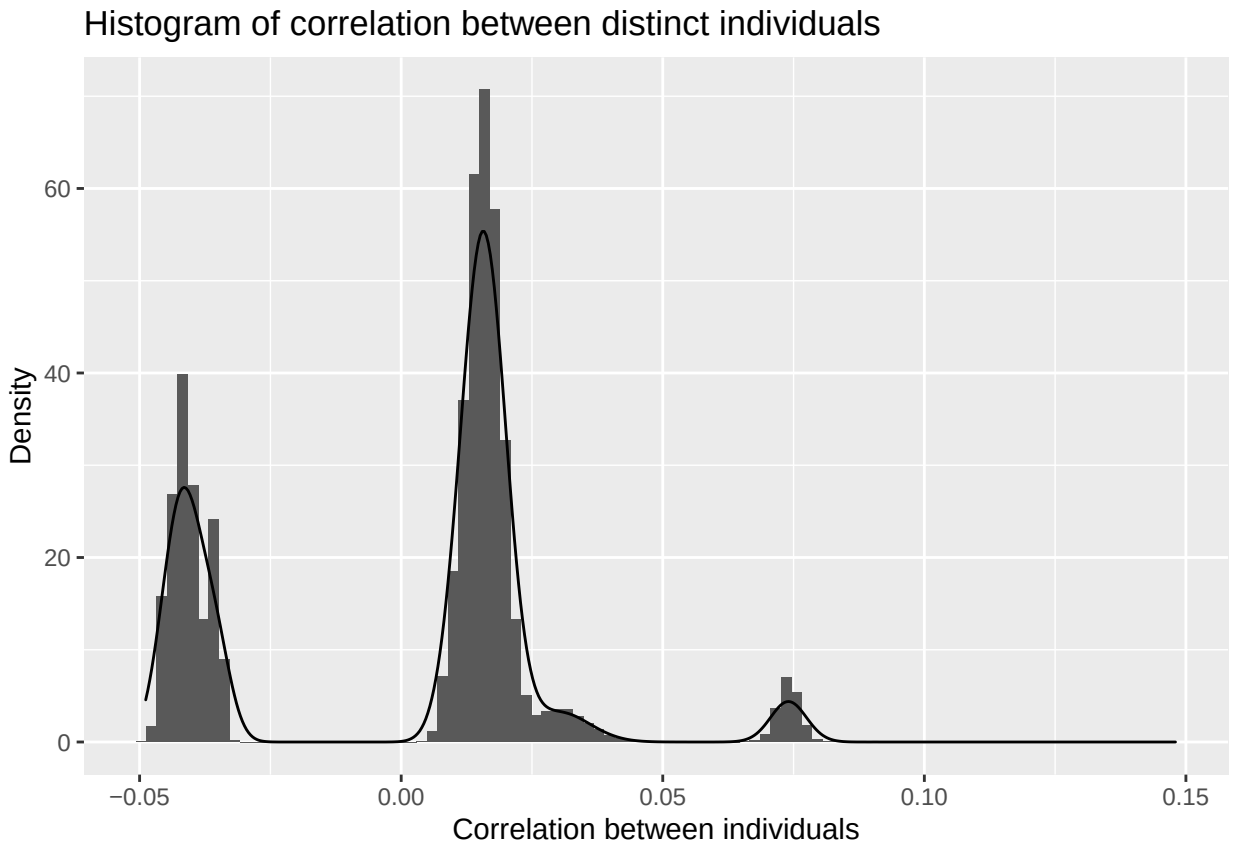
```
## Warning in which.snps & is.autosome(x@snps$chr): longer object length is not a
```

```
## multiple of shorter object length
```

```
# transform matrix into dataframe (3 columns - col1 = samples each row, col2 = samples each column ,  c
dfGrm = reshape2::melt(grm_alt_p)

# indexing with numeric values each sample (columns and rows)
# dfGrm$sampLines = rep ( seq ( 1 , nrow ( grm_alt_p ) ) , nrow ( grm_alt_p ) )
# dfGrm$sampCols = sort ( rep ( seq ( 1 , nrow ( grm_alt_p ) ) , nrow ( grm_alt_p ) ) )


# To calculate the correlation between individuals, the calculation A_ij/sqrt(A_ii)sqrt(A_jj) will be d
# dataframe with only diag. values
dfGrmDiag = dfGrm[ dfGrm$Var1 == dfGrm$Var2,]
# sqrt of those values
dfGrmDiag = dfGrmDiag %>% mutate ( sqrtVal = sqrt ( value ) , sqrtVal2 = sqrt ( value ) )

# merging each A_ii for each row and col
dfGrmM = merge ( dfGrm , dfGrmDiag[ ,c ( "sqrtVal" , "Var1" ) ] , on = c ( "Var1" ) )
dfGrmM2 = merge ( dfGrmM , dfGrmDiag[ ,c ( "sqrtVal2" , "Var2" ) ] , on = c ( "Var2" ) )

# Calculating A_ij/(sqrt(A_ii)sqrt(A_jj))
dfGrmFinal = dfGrmM2 %>% mutate ( corrIndividuals = value / ( sqrtVal * sqrtVal2 ) ) %>% arrange ( Var1

# plot heatmap - correlation between individuals
dfGrmFinal %>% ggplot( aes ( x = Var1 , y = Var2 , fill = corrIndividuals ) ) +
geom_tile() +
theme( axis.text.x = element_text(angle = 90, hjust = 1) , text = element_text (size = 5) ) +
labs ( x = "Sample ID" , y = "Sample ID" )
```




```
# It seems there are blocks with higher correlation between individuals between the samples

# Filter of all correlation values between individuals
dfUniqueCorr = dfGrmFinal %>% filter ( corrIndividuals < .9999 ) %>% distinct ( corrIndividuals , .keep

# Histogram and density of correlation values
dfUniqueCorr %>% ggplot ( aes ( x = corrIndividuals ) ) +
geom_histogram ( aes(y=..density..) , bins = 100 ) +
geom_density ( ) +
labs ( x = "Correlation between individuals" , y = "Density" , title = "Histogram of correlation betwee
```




```
# The correlation blocks are bolder in this plot

# Readind file with HLA expressions and ancestry information
hlaExp = readr::read_tsv("/raid/genevol/heritability/hla_expression.tsv")
```

```
## Parsed with column specification:
## cols(
##   subject_id = col_character(),
##   continental_pop = col_character(),
##   population = col_character(),
##   sex = col_character(),
##   gene_name = col_character(),
##   NumReads = col_double(),
##   TPM = col_double()
## )
```

```
# Ancestry of all samples
ancestry = unique ( hlaExp[ , c ( "subject_id" , "continental_pop" )] )

# Merging ancestry info with correlation dataframe
check = merge ( dfUniqueCorr , ancestry , by.x = c ( "Var1" ) , by.y = c ( "subject_id" ) )
check2 = merge ( check , ancestry , by.x = c ( "Var2" ) , by.y = c ( "subject_id" ) )

tableAncestry = unique ( check[,c("continental_pop" , "Var1")] ) %>% select ( continental_pop ) %>% tabl

knitr::kable( tableAncestry )
```

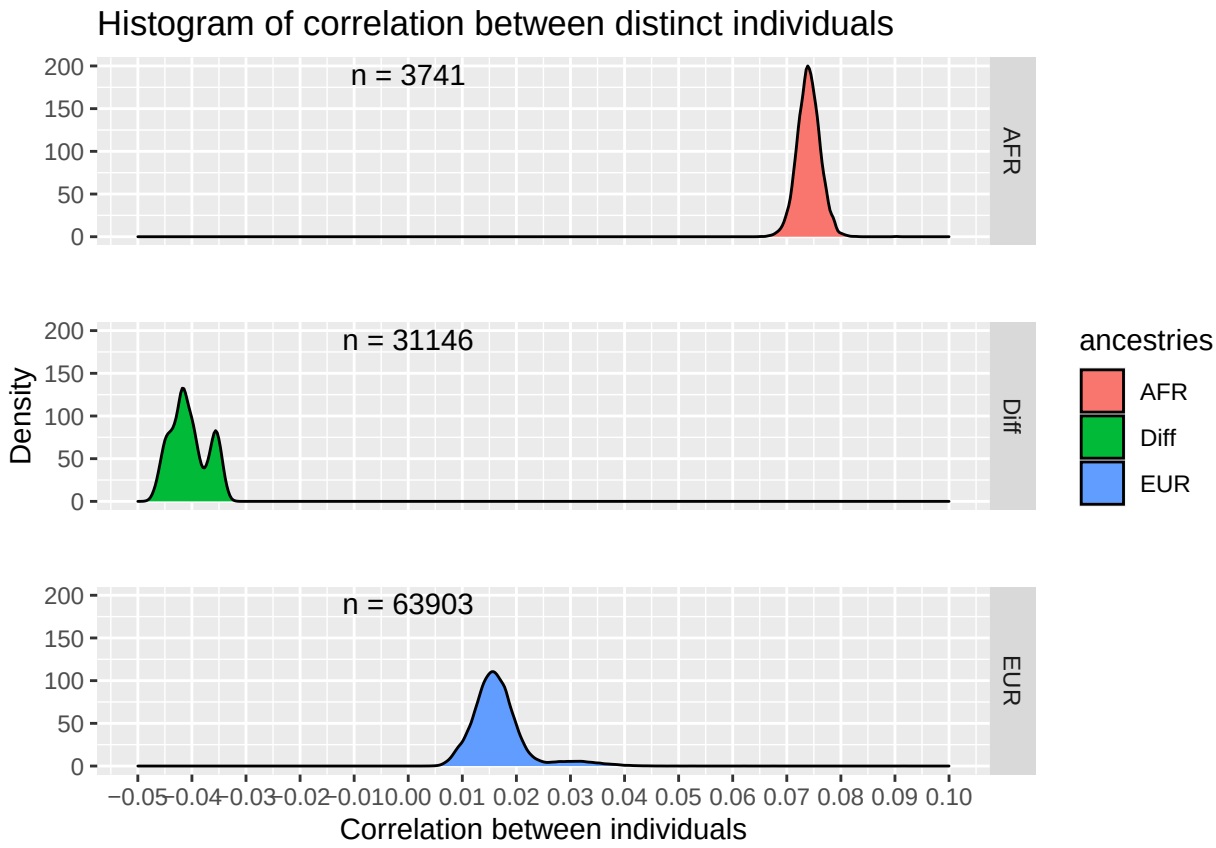
Ancestry	Freq	relFreq
AFR	87	19.59%
EUR	357	80.41%

```
# Approximately 20% of the 444 individuals are African, while the other 80% are European



# Checking the amount of comparisons between individuals with same ancestry and different ones
checkFin = check2 %>% mutate ( ancestries = ifelse ( continental_pop.x == continental_pop.y , continenta


tableComparisons = table ( checkFin$ancestries ) %>% as.data.frame() %>% mutate ( freqRel = Freq/ sum (

knitr::kable ( tableComparisons )
```

Ancestry	NumComparisons	freqRel
AFR	3741	0.0378682
Diff	31146	0.3152748
EUR	63903	0.6468570

```
numComparisons = plyr::ddply(.data=checkFin,
                  "ancestries",
                  summarize,
                  n=paste("n =", length(corrIndividuals)))

checkFin %>% ggplot ( aes ( x = corrIndividuals , fill =  ancestries ) ) +
    geom_density ( ) +
    facet_grid ( ancestries~. ) +
    theme(panel.spacing = unit (2, "lines") ) +
    labs ( x = "Correlation between individuals" , y = "Density" , title = "Histogram of correlation be
  geom_text(data=numComparisons, aes(x=0, y=190, label=n),
                    colour="black", inherit.aes=FALSE, parse=FALSE) +
  scale_x_continuous ( breaks = seq ( from = -0.05 , to = 0.1 , by = 0.01 ) , limits = c( -0.05 , 0.1 )
```

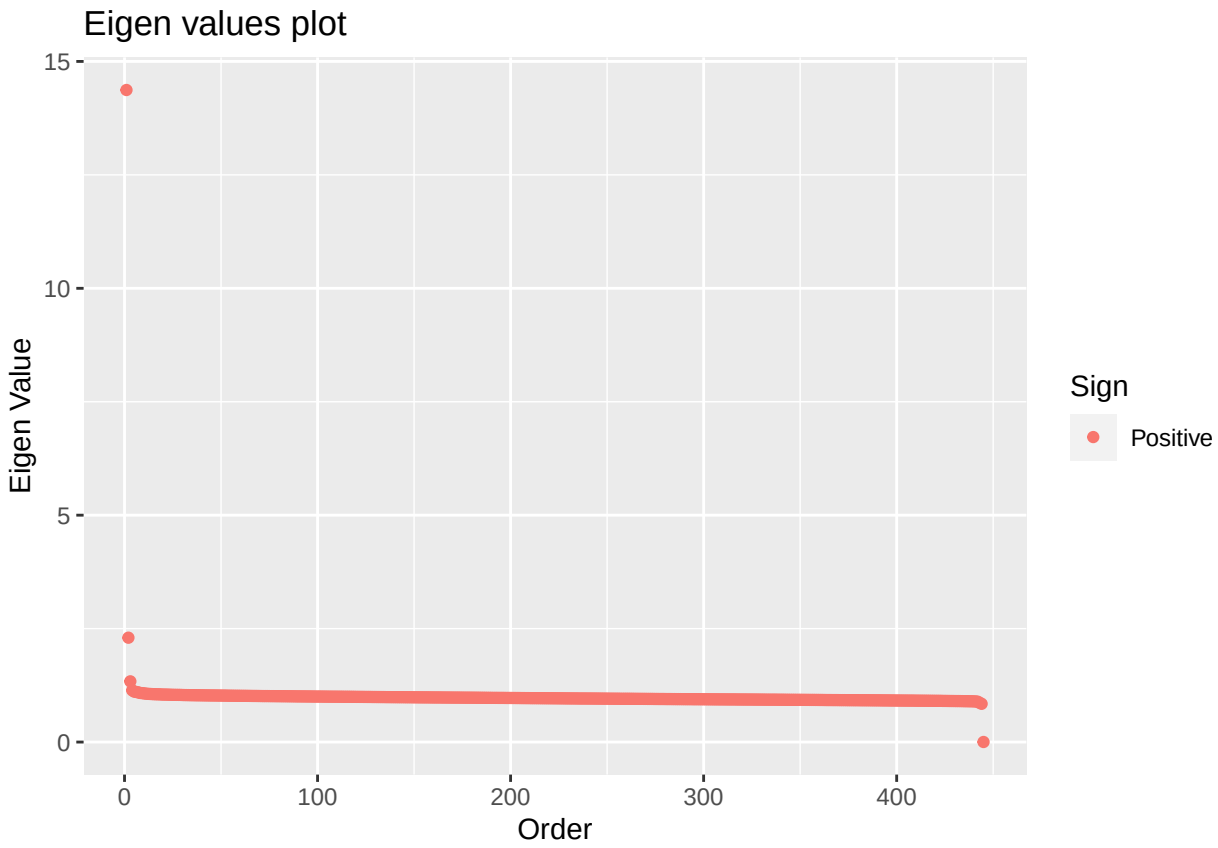
```
## Warning: Removed 2 rows containing non-finite values (stat_density).
```




```
# display individuals with correlation greater than 10% in the sample
listGreatCorr = checkFin[ ( checkFin$corrIndividuals > .1 ) & ( checkFin$corrIndividuals < .999 ) , ] %
```

```
listGreatCorr
```

```
# grm = grm_alt_p
# rownames ( grm ) = altReadSnps
# colnames ( grm ) = altReadSnps$sample.id

correlationMatrix = reshape2::dcast(dfGrmFinal, Var1~Var2 , value.var = "corrIndividuals")
rownames ( correlationMatrix ) = correlationMatrix$Var1
correlationMatrix$Var1 = NULL

eigenValuesGrm = eigen ( correlationMatrix )
dfEigen = eigenValuesGrm$values %>%
as.data.frame ( ) %>%
mutate ( order = 1:445 ) %>%
rename ( "Value" = '.' ) %>%
mutate ( neg = ifelse ( Value < 0 , "Negative" , "Positive" ) )


dfEigen %>% ggplot ( aes ( x = order , y = Value , colour = neg )  ) +
geom_point ( ) +
labs ( x = "Order" , y = "Eigen Value" , title = "Eigen values plot" , colour = "Sign" )
```

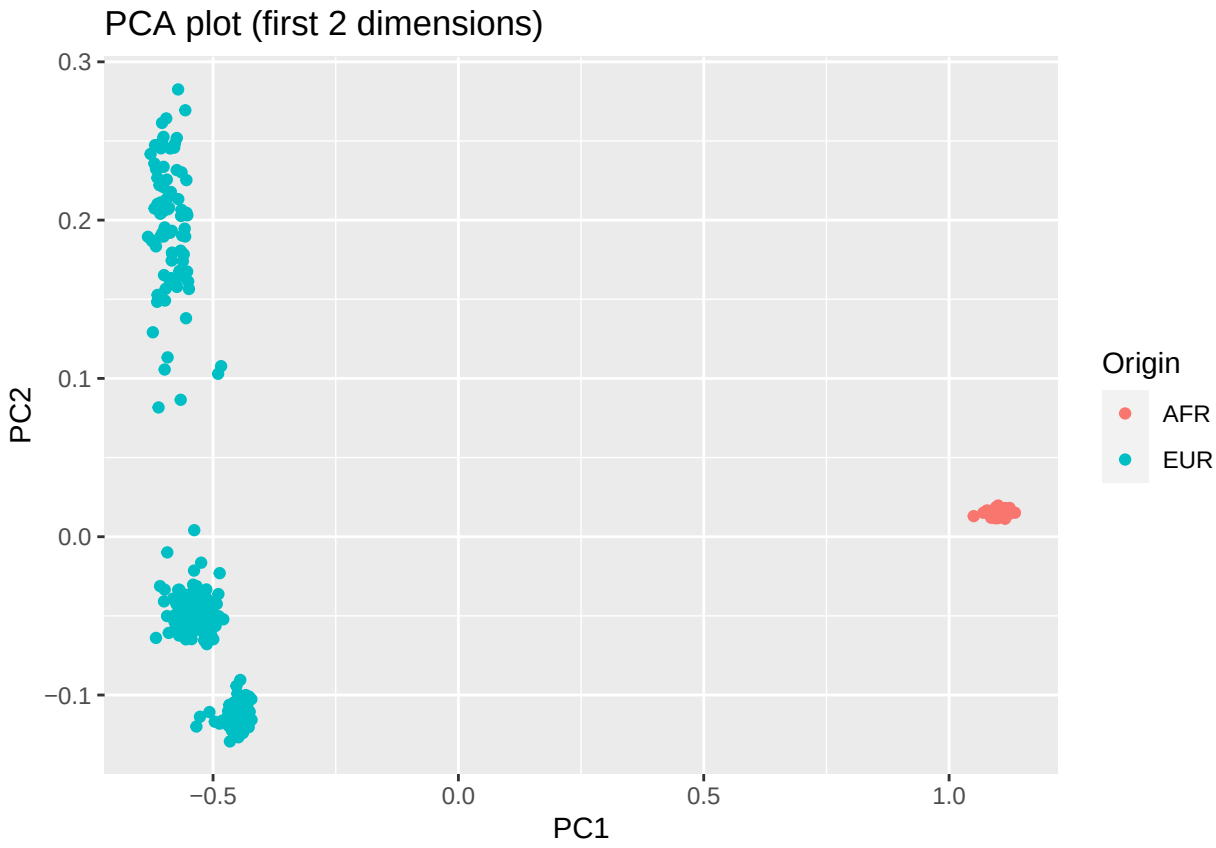



```
# matrixCorrection = eigenValuesGrm$vectors %*% diag( eigenValuesGrm$values + abs ( min ( eigenValuesGr
# rownames ( matrixCorrection ) = grm_obj$sample.id
# colnames ( matrixCorrection ) = grm_obj$sample.id

# rownames ( matrixCorrection ) = rownames ( grm )
# colnames ( matrixCorrection ) = colnames ( grm )


# eigenCorr = eigen ( matrixCorrection )

# dfEigenCorr = eigenCorr$values %>%
  # as.data.frame ( ) %>%
  # mutate ( order = 1:n() ) %>%
  # rename ( "Value" = '.' ) %>%
  # mutate ( neg = ifelse ( Value < 0 , "Negative" , "Positive" ) )


# dfEigenCorr %>% ggplot ( aes ( x = order , y = Value , colour = neg )  ) +
#   geom_point ( ) +
#   labs ( x = "Order" , y = "Eigen Value" , title = "Eigen values plot" , colour = "Sign" )
```

```
expressionInterest = hlaExp %>% filter ( subject_id %in% colnames ( correlationMatrix ) )

mainInfo = expressionInterest %>% distinct( subject_id , continental_pop ,population )
numEigen = 2
print ( paste0 ( "Total variation explained by the first ", numEigen , " eigen values: " , 100*round ( s
```

```
## [1] "Total variation explained by the first 2 eigen values: 3.75%"
```

```
vectors_ = eigenValuesGrm$vectors[,1:numEigen]
calcScores = as.matrix ( correlationMatrix , ncol = 445 )  %*% vectors_ %>%
  as.data.frame() %>%
  rename ( "PC1" = "V1" , "PC2" = "V2" ) %>%
  mutate ( subject_id = rownames ( correlationMatrix ) )
pcaPlot = merge ( mainInfo , calcScores )

pcaPlot %>% ggplot ( aes ( x = PC1 , y = PC2  , colour = continental_pop ) ) +
  geom_point ( ) +
  labs ( title = "PCA plot (first 2 dimensions)" , colour = "Origin" )
```




```
  # geom_text ( )
```

```
simpleModels = function ( exp_ , df ){

  dfFilter = df  %>% filter ( gene_name == exp_ )

  fixed0 = lm ( TPM ~ 1 , data = dfFilter )
  sum0 = summary ( fixed0 )
  fixedEffectSigma = sum0$sigma^2

  mixedModel = coxme::lmekin( dfFilter$TPM ~ 1 + (1|dfFilter$subject_id) , data=dfFilter, varlist=list(a

  mixedEffectSigma = mixedModel$sigma^2
  sigmaA = as.numeric(mixedModel$vcoef)

  # comparison = mixedEffectSigma/fixedEffectSigma
```

```
  # h = sigmaA / ( sigmaA + mixedEffectSigma)


  modelExpanded = coxme::lmekin( dfFilter$TPM ~ 1 + dfFilter$PC1 + dfFilter$PC2 + (1|dfFilter$subject_id

  mixedEffectSigmaExp <- modelExpanded$sigma^2
  # comparisonExp = modelExpanded/fixedEffectSigma
  sigmaAExp = as.numeric(modelExpanded$vcoef)

  # hExp = sigmaAExp / (sigmaAExp + mixedEffectSigmaExp )

  return ( c ( exp_ , fixedEffectSigma , mixedEffectSigma , sigmaA , mixedEffectSigmaExp , sigmaAExp ) )

}

listNames = unique ( expressionInterest$gene_name )
modelDf = merge ( expressionInterest , calcScores )

requiredInfo = NULL
for ( name_ in listNames ){

  requiredInfo = rbind ( requiredInfo , simpleModels ( exp_ = name_ ,df = modelDf ) )

}

finalDf = requiredInfo %>% as.data.frame ( ) %>% rename ("Gene" = "V1" ,
                                                         "fixedSigma" = "V2" ,
                                                         "residualMixedSigma" = "V3" ,
                                                         "randomEffectSigma" = "V4",
                                                         "residualMixedSigmaExp" = "V5" ,
                                                         "randomEffectSigmaExp" = "V6") %>%
mutate ( fixedSigma = as.numeric ( as.character ( fixedSigma ) ) ,
    residualMixedSigma = as.numeric ( as.character (residualMixedSigma)) ,
    randomEffectSigma =  as.numeric ( as.character (randomEffectSigma)) ,
    residualMixedSigmaExp = as.numeric ( as.character (residualMixedSigmaExp)),
    randomEffectSigmaExp = as.numeric ( as.character (randomEffectSigmaExp))
    ) %>%
  mutate ( comparisonNull = residualMixedSigma/fixedSigma ,
           comparisonNullExp = residualMixedSigmaExp/fixedSigma ,
           hSimple = randomEffectSigma / ( randomEffectSigma + residualMixedSigma ) ,
           hExpanded = randomEffectSigmaExp / ( randomEffectSigmaExp + residualMixedSigmaExp ))

finalDf %>% knitr::kable()
```

Gene	fixedSigma	residualMixedSigma	randomEffectSigma	residualMixedSigmaExp	randomEffectSigmaExp	comparisonNull	comparisonNullExp	hSimple	hExpanded
HLA-A	180900.1	5.0000000	3.145874e+10	0.000000e+00	2.913656e+07	0.00e+00	0.0000000	1.0000000	1.0000000
HLA-B	526383.0	5.0000000	7.950933e+18	7.000000e-07	5.286063e+05	0.00e+00	0.0000000	1.0000000	1.0000000

Gene	fixedSigma	residualMixedSigma	randomEffectSigma	residualMixedSigmaExp	randomEffectSigmaExp	comparisonNull	comparisonNullExp	hSimple	hExpanded
HLA-C	127438.40	0.0000000	1.522734e+21	1.240047e+05	7.945000e-04	0.00e+00	0.9730559	1.0000000	0.0000000
HLA-DPA1	31587.29	0.2019869	3.131345e+00	0.000000e+00	Inf	6.40e-06	0.0000000	0.9999935	NaN
HLA-DPB1	37625.10	0.5107438	3.385689e+04	6.598602e-01	3.329820e+04	1.36e-05	0.0000175	0.9999849	0.9999802
HLA-DQA1	51654.46	0.6312459	5.005205e+00	0.000000e+00	6.642456e+06	1.22e-05	0.0000000	0.9999871	1.0000000
HLA-DQB1	38524.93	0.8315939	3.883568e+04	3.683984e-01	3.880857e+04	2.16e-05	0.0000096	0.9999786	0.9999905
HLA-DRA	390199.56	2.1355405	3.821958e+05	1.640958e+00	3.799944e+05	1.57e-05	0.0000119	0.9999839	0.9999878
HLA-DRB1	148507.21	1.8322019	1.301254e+05	3.014181e+00	1.275788e+05	1.23e-05	0.0000203	0.9999859	0.9999764