handin2_part1.RMD

Stefano Pellegrini

9/30/2020

```
library(tidyverse)
```

1 Genotype calling based on genotype likelihoods

```
# Each individual has 3 cols: GL AA, GL AB, GL BB
# Number in allele1 and allele2 are the bases: A=0, C=1, G=2, T=3
# 50000 rows = SNPs
# 303 columns = SNP_ID + 2xalleles + 100individuals x 3GL
geno_lik <- read.table("input", header = TRUE)
geno_lik[1:5,1:6]
```

```
##      marker allele1 allele2 Ind0   Ind0.1   Ind0.2
## 1  1_846808       3       1    0 0.001949 0.998051
## 2  1_884815       0       2    0 0.000002 0.999998
## 3  1_927309       3       1    0 0.000001 0.999999
## 4  1_989461       0       2    0 0.000488 0.999512
## 5 1_1021415       0       2    0 0.015384 0.984616
```

```
dim(geno_lik)
```

```
## [1] 50000   303
```

```
# Rows are individuals same order as the columns of SNP (from fourth column)
pop <- read.table("pop.info")
colnames(pop) <- c("pop", "ID")
head(pop)
```

```
##   pop      ID
## 1 ASW NA19818
## 2 MXL NA19678
## 3 MXL NA19663
## 4 ASW NA19901
## 5 MXL NA19660
## 6 CHB NA18546
```

```
dim(pop)
```

```
## [1] 100   2
```

```
# True genotypes will be used to estimate accuracy of the methods
true_geno <- read.table("input.geno")
true_geno[1:5,1:3]
```

```
##           NA19818 NA19678 NA19663
```

```
## rs4475691       2       2       2
## rs4246503       2       2       2
## rs2341362       2       2       2
## rs9778201       2       2       2
## rs3737728       2       2       1
```

```
dim(true_geno)
```

```
## [1] 50000   100
```

1.2 Assignment

1.2.1 Estimate allele frequencies and ancestry proportions

The individuals are admixed so we cannot use the normal EM-algorithm to obtain allele frequencies therefore we need to use NGSadmix which will estimate both admixture proportions and allele frequencies for each ancestral population. Rerun the NGSadmix analysis assuming 3 ancestral populations without filtering away SNPs and using a seed (-s) with the following command.

```
NGSadmix=/data/albrecht/advBinf/prog/angsd/misc/NGSadmix
$NGSadmix -likes /data/albrecht/advBinf/ngsAdmix/input.gz -K 3 -P 3 -minMaf 0 -o assign3 -s 1
```

Identify the ancestral populations of each of the 3 columns (same in both les) using the inferred admixture proportions and the population information file. Which columns represents Africa, European and Asian?

```
echo ADMixture proportion
cat Data/part1/assign3.qopt | head -10 | tail -5
echo Populations
cat Data/part1/pop.info | head -10 | tail -5
```

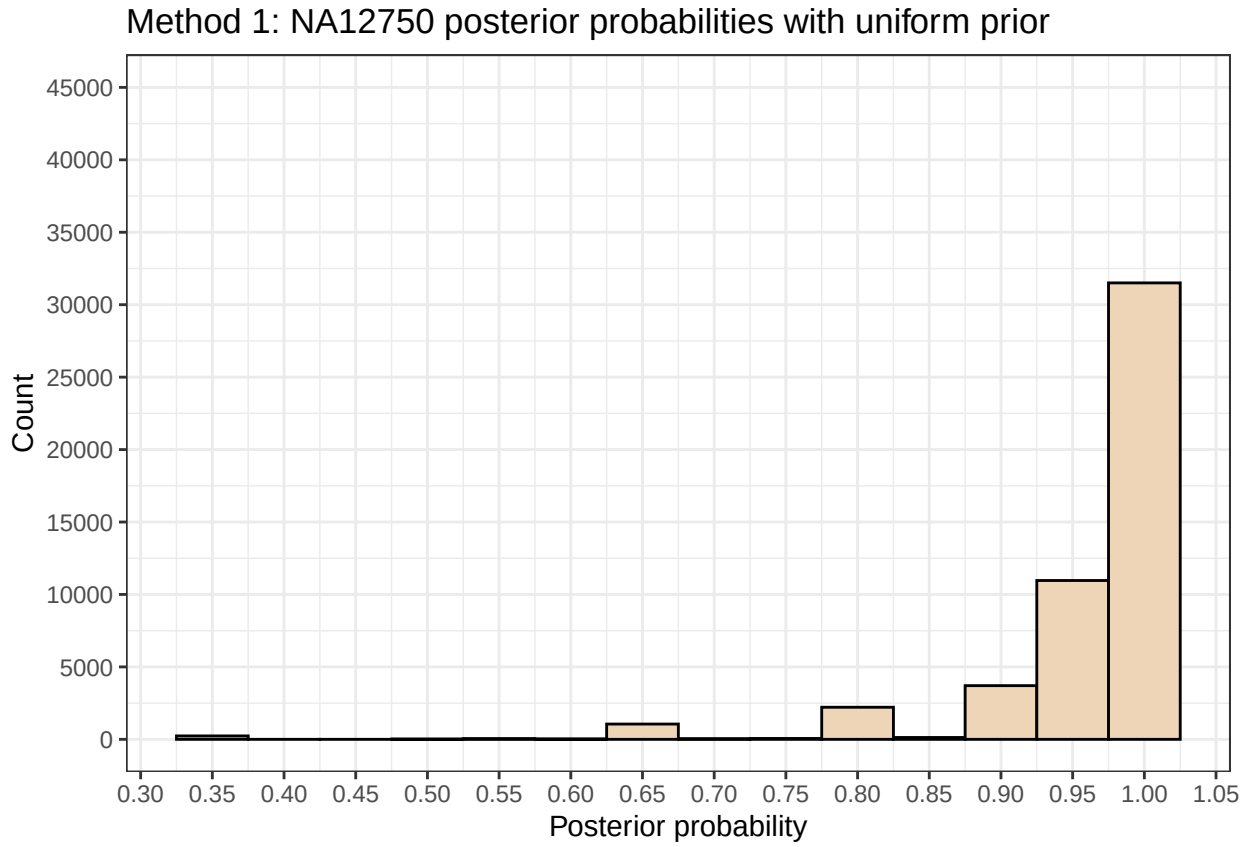
Column 1 rapresents africans, column 2 rapresents europeans, and column 3 rapresents asians.

1.2.2 Call genotypes using different methods

Write code to call the 50,000 genotypes for NA12750 assuming a uniform prior for the 3 possible genotypes and calculate the posterior probability for each called genotype. Plot a histogram of the posterior probabilities.

```
# Method 1: uniform priors
get_post_uPrior <-  function(df){
  colnames(df)[3:5] <- c("Ind", "Ind.1", "Ind.2")
  df %>%
    # Assign uniform prior (the posterior will be equal to the genotype likelihood)
    mutate(gf = 1/3,
           gf.1 = 1/3,
           gf.2 = 1/3) %>%
    # Compute genotype posterior probabilities
    mutate(den_post = Ind*gf + Ind.1*gf.1 + Ind.2*gf.2,
           post = Ind*gf / den_post,
           post.1 = Ind.1*gf.1 / den_post,
           post.2 = Ind.2*gf.2 / den_post) %>%
    # Call genotype for each SNP
    mutate(post_max = apply(.[,c("post", "post.1", "post.2")], 1, function(x) max(x))) %>%
    mutate(geno_call = apply(.[,c("post", "post.1", "post.2")], 1, function(x) which.max(x)-1)) %>%
    # Comment next line for debugging
```

```
    select(starts_with("post"), "geno_call") -> df
  return(df)
}

NA12750_geno <- geno_lik %>% select(starts_with("allele"), starts_with("Ind60"))
NA12750_geno_uniform <- get_post_uPrior(df = NA12750_geno)

# Plot histogram
NA12750_geno_uniform %>%
  ggplot(aes(post_max)) + geom_histogram(col="black", fill="bisque2", binwidth = 0.05) +
  labs(title = "Method 1: NA12750 posterior probabilities with uniform prior") +
  ylab("Count") +
  xlab("Posterior probability") +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10), limits = c(0,45000)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 20)) +
  theme_bw()
```



For the same individual write code to call genotypes using the frequency as a prior assuming Hardy-Weinberg equilibrium. Write your choice of frequency and plot a histogram of the posterior probabilities.

```
# Method 2: HWE (CEU) genotype frequencies as priors
freq_pop <- read.table("assign3.fopt")
colnames(freq_pop) <- c("freq_A1", "freq_A2", "freq_A3")
admix <- read.table("assign3.qopt")
```

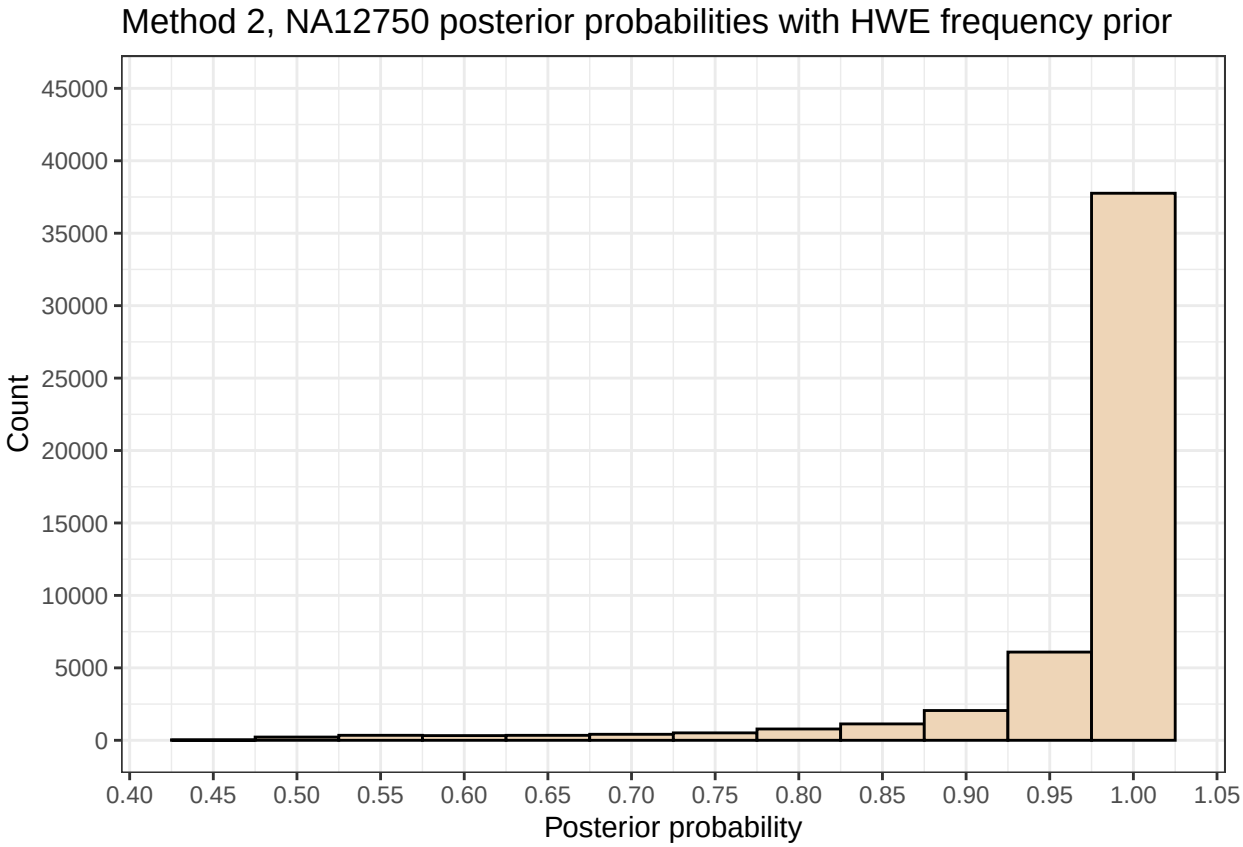
```r
# Use each allele freq as a prior to compute the posterior of the genotypes
get_post_fPrior <-  function(df, A_freq){
  colnames(df)[3:5] <- c("Ind", "Ind.1", "Ind.2")
  df <- cbind(df, Afreq = A_freq)
  df %>%
    # Compute genotypes frequencies (priors)
    mutate(gf = (1 - Afreq)^2,
           gf.1 = 2*(1 - Afreq)*Afreq,
           gf.2 = (Afreq)^2) %>%
    # Compute genotype posterior probabilities
    mutate(den_post = Ind*gf + Ind.1*gf.1 + Ind.2*gf.2,
           post = Ind*gf / den_post,
           post.1 = Ind.1*gf.1 / den_post,
           post.2 = Ind.2*gf.2 / den_post) %>%
    # Call genotype for each SNP
    mutate(post_max = apply(.[,c("post", "post.1", "post.2")], 1, function(x) max(x))) %>%
    mutate(geno_call = apply(.[,c("post", "post.1", "post.2")], 1, function(x) which.max(x)-1)) %>%
    # Comment next line for debugging
    select(starts_with("post"), "geno_call") -> df
  return(df)
}

# Check ancestral population
admix[61,]
```

```
##       V1 V2           V3
## 61 1e-09  1 3.344077e-08
```

```r
pop[61,]
```

```
##    pop      ID
## 61 CEU NA12750
```

```r
# Use european ancestral population frequencies as priors
NA12750_geno_A2_priors <- get_post_fPrior(df = NA12750_geno, A_freq = freq_pop$freq_A2)

# Plot histogram
NA12750_geno_A2_priors %>%
  ggplot(aes(post_max)) + geom_histogram(col="black", fill="bisque2", binwidth = 0.05) +
  labs(title = "Method 2, NA12750 posterior probabilities with HWE frequency prior") +
  ylab("Count") +
  xlab("Posterior probability") +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10), limits = c(0,45000)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 20)) +
  theme_bw()
```



The MAF of european population is used to compute the genotypes frequencies, which are used as priors to compute the genotype posterior probabilities of individual NA12750.

Perform haplotype imputation using BEAGLE (the input.gz is in the beagle input format) on the same data and call genotypes based on the maximum posterior probability (see exercise for example). For the same individual extract the posterior probability as well and make a histogram.

```
# Main folder
FOLDER=/isdata/othergrp/ida/advBinf
# Beagle
BEAGLE=$FOLDER/prog/beagle.jar
# Run the imputation based on the genotype likelihoods
java -jar $BEAGLE like=input.gz out=imputation
```

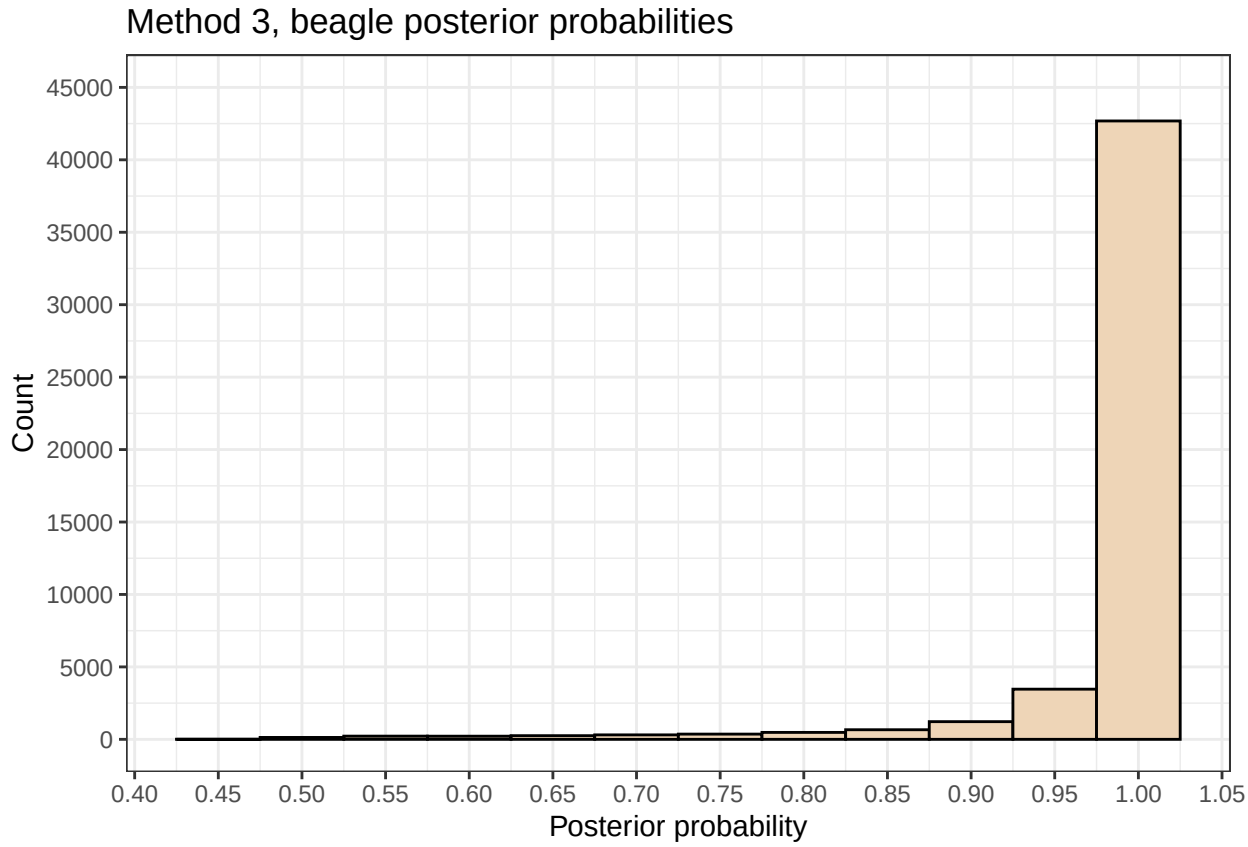
```
# Method 3: beagle posteriors probabilities
beagle_geno <- read.table("imputation.input.gz.gprobs", header=TRUE)
beagle_geno[1:5,1:6]
```

```
##      marker alleleA alleleB Ind0 Ind0.1 Ind0.2
## 1  1_846808       3       1    0 0.0009 0.9991
## 2  1_884815       0       2    0 0.0000 1.0000
## 3  1_927309       3       1    0 0.0000 1.0000
## 4  1_989461       0       2    0 0.0001 0.9999
## 5 1_1021415       0       2    0 0.0026 0.9974
```

```
dim(beagle_geno)
```

```
## [1] 50000   303
```

```
# Get the (higher) posterior probability of beagle called genotype
NA12750_geno_beagle <- beagle_geno %>% select(starts_with("Allele"), starts_with("Ind60")) %>%
  mutate(post_max = apply(.[,c("Ind60", "Ind60.1", "Ind60.2")], 1, function(x) max(x))) %>%
  # Call genotype for each SNP
  mutate(geno_call = apply(.[,c("Ind60", "Ind60.1", "Ind60.2")], 1, function(x) which.max(x)-1))

# Plot histogram
NA12750_geno_beagle %>%
  ggplot(aes(post_max)) + geom_histogram(col="black", fill="bisque2", binwidth = 0.05) +
  labs(title = "Method 3, beagle posterior probabilities") +
  ylab("Count") +
  xlab("Posterior probability") +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10), limits = c(0,45000)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 20)) +
  theme_bw()
```



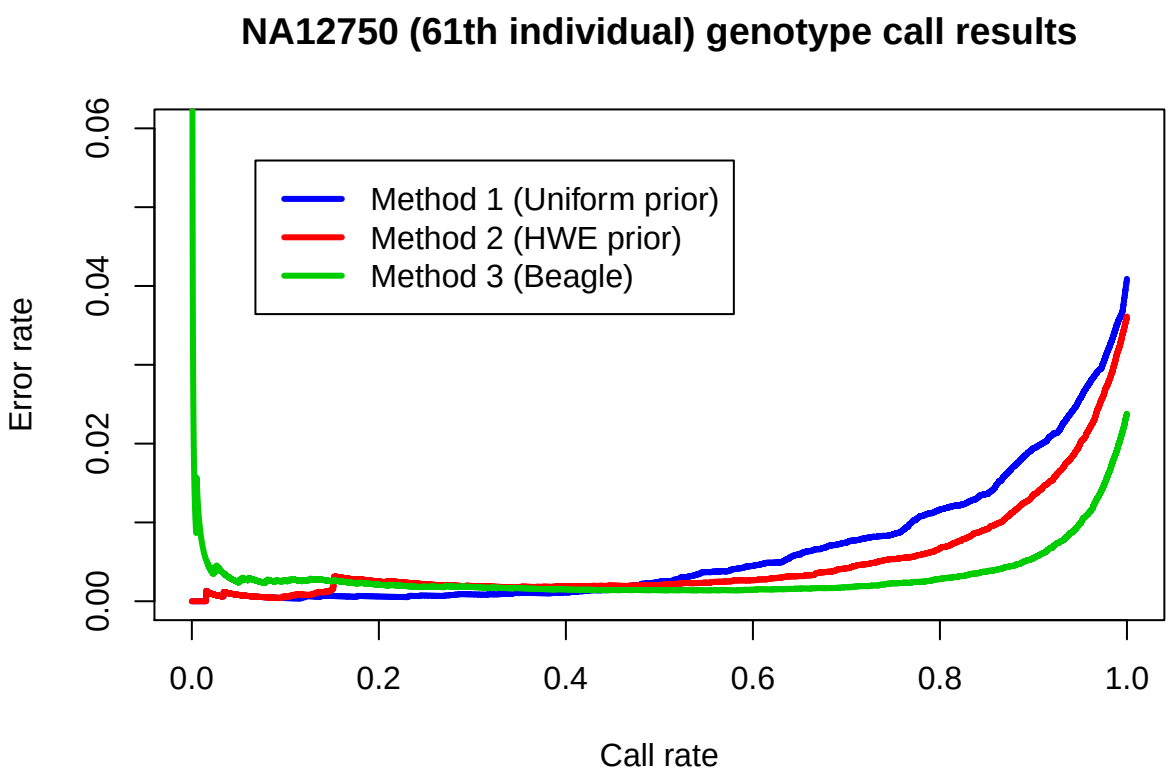
1.2.3 Evaluation

Use the true genotypes (input.geno file) to compare with your genotype calls.

Make a plot with the accuracy of the three genotyping approaches.

```
# Function to plot
plotAccuracy <- function(x, p, ...){
  p <- p[!is.na(x)]
```

```
  x <- x[!is.na(x)]
  ord <- order(p, decreasing = T)
  lines(1: length(x) / length(x), cumsum(!x[ord]) / 1:length(x), ...)
}

# Method 1 (uniform prior)
PP1 <- NA12750_geno_uniform$post_max
CP1 <- NA12750_geno_uniform$geno_call == true_geno[,"NA12750"]
# Method 2 (HWE prior)
PP2 <- NA12750_geno_A2_priors$post_max
CP2 <- NA12750_geno_A2_priors$geno_call == true_geno[,"NA12750"]
# Method 3 (beagle)
PP3 <- NA12750_geno_beagle$post_max
CP3 <- NA12750_geno_beagle$geno_call == true_geno[,"NA12750"]

# Plot accuracy
plot(1, xlim = 0:1, ylim = c(0, 0.06),
     col = "transparent",
     xlab = "Call rate",
     ylab = "Error rate",
     main = "NA12750 (61th individual) genotype call results")
plotAccuracy(CP1, PP1, lwd = 3, col = "blue")
plotAccuracy(CP2, PP2, lwd = 3, col = "red")
plotAccuracy(CP3, PP3, lwd = 3, col = "green3")
legend("topleft", inset=.1,
       legend=c("Method 1 (Uniform prior)", "Method 2 (HWE prior)", "Method 3 (Beagle)"),
       col=c("blue", "red", "green3"), lwd = 3)
```



You should then try to make a similar analysis for the third individual (NA19663, Ind2).

Try to use all 3 ancestral population frequencies as a prior (one at a time).

```
# Method 1: all 3 ancestral population frequencies priors
NA19663_geno <- geno_lik %>% select(starts_with("allele"), Ind2, Ind2.1, Ind2.2)

# Check the ancestral population
admix[3,]
```

```
##          V1        V2        V3
## 3 0.0125438 0.5926066 0.3948496
```

```
pop[3,]
```

```
##   pop      ID
## 3 MXL NA19663
```

```
# Use each allele freq as a prior to compute the posterior of the genotypes
NA19663_geno_A1_priors <- get_post_fPrior(df = NA19663_geno, A_freq = freq_pop$freq_A1)
NA19663_geno_A2_priors <- get_post_fPrior(df = NA19663_geno, A_freq = freq_pop$freq_A2)
NA19663_geno_A3_priors <- get_post_fPrior(df = NA19663_geno, A_freq = freq_pop$freq_A3)
```

Try to make the optimal prior using a combinations of the 3 ancestral frequencies.

```
# Method 2a: Optimal prior using weighted average genotype frequency with admixture proportion
get_post_wAvg_fPrior <-  function(df, A_all_freq, ad){
  colnames(df)[3:5] <- c("Ind", "Ind.1", "Ind.2")
```

```
  df <- cbind(df, A_all_freq)
  df %>%
    # Compute the weighted genotypes frequencies (priors)
    mutate(freq_A1 = freq_A1*ad[1,1],
           freq_A2 = freq_A2*ad[1,2],
           freq_A3 = freq_A3*ad[1,3]) %>%
    mutate(gf1 = (1 - freq_A1)^2,
           gf1.1 = 2*(1 - freq_A1)*freq_A1,
           gf1.2 = (freq_A1)^2,
           gf2 = (1 - freq_A2)^2,
           gf2.1 = 2*(1 - freq_A2)*freq_A2,
           gf2.2 = (freq_A2)^2,
           gf3 = (1 - freq_A3)^2,
           gf3.1 = 2*(1 - freq_A3)*freq_A3,
           gf3.2 = (freq_A3)^2,) %>%
    # Get weighted average genotypes frequencies (combined priors)
    mutate(gf = (gf1 + gf2 + gf3) / 3,
           gf.1 = (gf1.1 + gf2.1 + gf3.1) / 3,
           gf.2 = (gf1.2 + gf2.2 + gf2.2) / 3) %>%
    # Compute genotype posterior
    mutate(den_post = Ind*gf + Ind.1*gf.1 + Ind.2*gf.2,
           post = Ind*gf / den_post,
           post.1 = Ind.1*gf.1 / den_post,
           post.2 = Ind.2*gf.2 / den_post) %>%
    # Call genotype for each SNP
    mutate(post_max = apply(.[,c("post", "post.1", "post.2")], 1, function(x) max(x))) %>%
    mutate(geno_call = apply(.[,c("post", "post.1", "post.2")], 1, function(x) which.max(x)-1)) %>%
    # Comment next line for debugging
    select(starts_with("post"), "geno_call") -> df
  return(df)
}

NA19663_geno_wAvg_priors_a <- get_post_wAvg_fPrior(df = NA19663_geno,
                                                   A_all_freq = freq_pop,
                                                   ad = admix[3,])
```

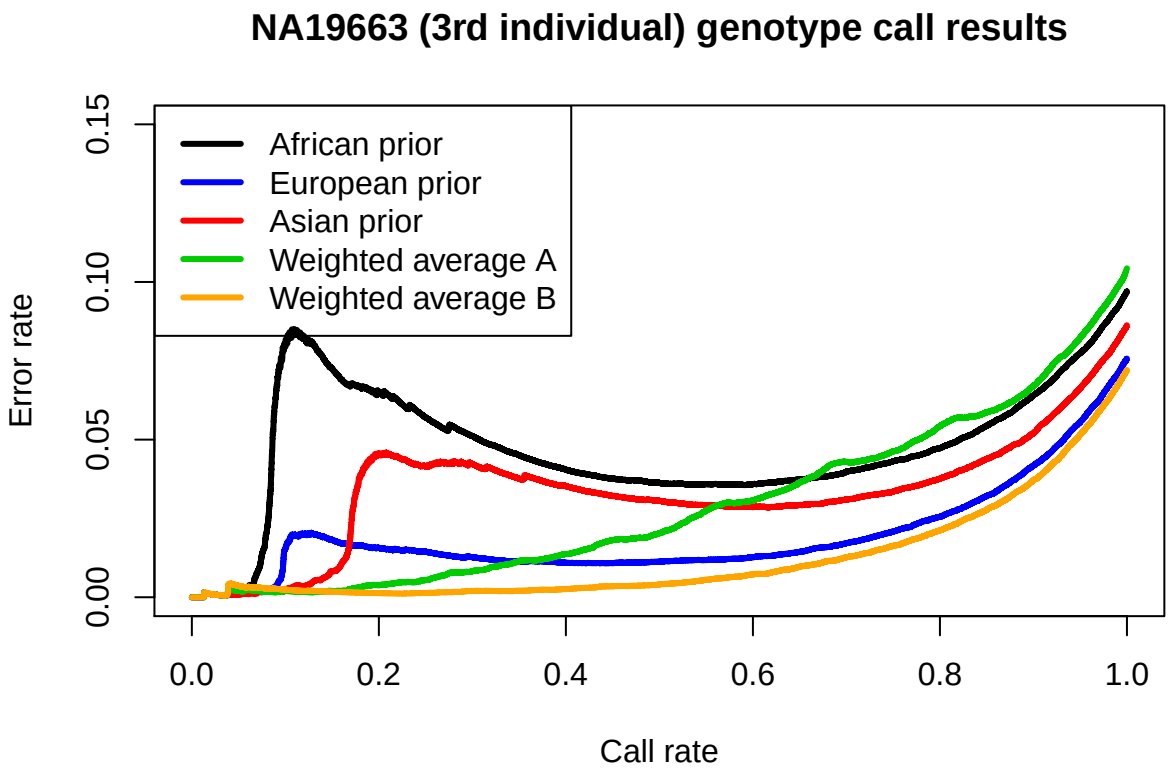
```
# Method 2b: Weighting the genotype likelihood with admixture proportion
get_post_wAvg_fPrior <-  function(df, A_all_freq, ad){
  colnames(df)[3:5] <- c("Ind", "Ind.1", "Ind.2")
  df <- cbind(df, A_all_freq)
  df %>%
    # Compute genotypes frequencies (priors)
    mutate(gf1 = (1 - freq_A1)^2,
           gf1.1 = 2*(1 - freq_A1)*freq_A1,
           gf1.2 = (freq_A1)^2,
           gf2 = (1 - freq_A2)^2,
           gf2.1 = 2*(1 - freq_A2)*freq_A2,
           gf2.2 = (freq_A2)^2,
           gf3 = (1 - freq_A3)^2,
           gf3.1 = 2*(1 - freq_A3)*freq_A3,
           gf3.2 = (freq_A3)^2,) %>%
    # Get weighted average genotypes frequencies (combined priors)
    mutate(gf = (gf1*ad[1,1] + gf2*ad[1,2] + gf3*ad[1,3]) / 3,
           gf.1 = (gf1.1*ad[1,1] + gf2.1*ad[1,2] + gf3.1*ad[1,3]) / 3,
```

```
          gf.2 = (gf1.2*ad[1,1] + gf2.2*ad[1,2] + gf2.2*ad[1,3]) / 3) %>%
    # Compute genotype posterior
    mutate(den_post = Ind*gf + Ind.1*gf.1 + Ind.2*gf.2,
          post = Ind*gf / den_post,
          post.1 = Ind.1*gf.1 / den_post,
          post.2 = Ind.2*gf.2 / den_post) %>%
    # Call genotype for each SNP
    mutate(post_max = apply(.[,c("post", "post.1", "post.2")], 1, function(x) max(x))) %>%
    mutate(geno_call = apply(.[,c("post", "post.1", "post.2")], 1, function(x) which.max(x)-1)) %>%
    # Comment next line for debugging
    select(starts_with("post"), "geno_call") -> df
  return(df)
}

NA19663_geno_wAvg_priors_b <- get_post_wAvg_fPrior(df = NA19663_geno,
                                                   A_all_freq = freq_pop,
                                                   ad = admix[3,])
```

The optimal prior was obtained by computing a weighted average of the genotype frequencies, using the AD mixture proportions as weights.

Plot the results in a single plot.

```
# A1 population (africans)
PP1 <- NA19663_geno_A1_priors$post_max
CP1 <- NA19663_geno_A1_priors$geno_call == true_geno[,"NA19663"]
# A2 population (european)
PP2 <- NA19663_geno_A2_priors$post_max
CP2 <- NA19663_geno_A2_priors$geno_call == true_geno[,"NA19663"]
# A3 population (asians)
PP3 <- NA19663_geno_A3_priors$post_max
CP3 <- NA19663_geno_A3_priors$geno_call == true_geno[,"NA19663"]
# Optimal prior (weighted average)
PP4a <- NA19663_geno_wAvg_priors_a$post_max
CP4a <- NA19663_geno_wAvg_priors_a$geno_call == true_geno[,"NA19663"]
PP4b <- NA19663_geno_wAvg_priors_b$post_max
CP4b <- NA19663_geno_wAvg_priors_b$geno_call == true_geno[,"NA19663"]

# Plot accuracy
plot(1, xlim = 0:1, ylim = c(0, 0.15),
     col = "transparent",
     xlab = "Call rate",
     ylab = "Error rate",
     main = "NA19663 (3rd individual) genotype call results")
plotAccuracy(CP1, PP1, lwd = 3, col = "black")
plotAccuracy(CP2, PP2, lwd = 3, col = "blue")
plotAccuracy(CP3, PP3, lwd = 3, col = "red")
plotAccuracy(CP4a, PP4a, lwd = 3, col = "green3")
plotAccuracy(CP4b, PP4b, lwd = 3, col = "orange")
legend("topleft",
       legend=c("African prior", "European prior", "Asian prior", "Weighted average A", "Weighted averag
       col=c("black", "blue", "red", "green3", "orange"), lwd = 3)
```


NA19663 (3rd individual) genotype call results
African prior
European prior
Asian prior
Weighted average A
Weighted average B
Error rate
0.00
0.05
0.10
0.15
0.0
0.2
0.4
0.6
0.8
1.0
Call rate