Capstone HLAofEthnics

Jozsef Toth

2024-07-11

Introduction

The main goal of this project is to develop a lightweight model for predicting the patient's ancestry (descent) by using the subject's human MHC class I (Major Histocompatibility Complex class I) alleles. For this project, I choose the Asian and European ancestry, because these two have the most data in the IPD-IMGT/HLA database with the proper HLA details. First we download the data by using HTTP requests and json parser, than we clean the data. After applying different approaches we will see that we can reach more than 90% accuracy for this 2 descents. The used methods are Classification tree,KNN, GLM and a self developed model with high scalability and speed.

Background

The human's have 6 MHC class I molecules, 2 HLA-A and 2 HLA-B and 2 HLA-C which are inherited from the parents, one A,B,C from each. These molecules are essential for the adaptive immune system because they help to identify and kill the viruses and some kind of bacteria. These alleles and the related epitope binding tools are commonly used in the vaccine design process to choose better targets and create population based or personalized vaccines. Like most of the prediction tools we will focus on the 4 digit alleles.

Data download / Preparation

The data will be directly downloaded from the IPD-IMGT/HLA API. First we will get all the available cell IDs (There are 68658 at July 15,2024) than make a http request for each of them. This can take a long time, around 10 hours, so i highly recommend to use the pre downloaded dataset what is included in this project (imgt_cell_dat_raw.csv)

Helper functions We will use the convert_4_digit_allele function to keep only the 4 digit alleles. The 2 digit alleles and missing alleles will be converted to NA, and the more detailed alleles will be cut at 4 digits. The ConvertNull function is only a helper to deal with the missing properties.

```
################################################################
# Download data from IPD-IMGT/HLA
# Download time is pretty long, about 10 hours,
# Please use the prepared imgt_cell_dat_raw.csv from Github
################################################################

#Helper function to handle null values
ConvertNull <- function(obj) {
  ifelse(is.null(obj),NA,obj)
}
```

```r
#Helper function to convert allele digits to 4 digit format
convert_4_digit_allele <- function(s) {
  result <- NA
  if(!is.na(s)) {
    s_parts <- unlist(strsplit(s, split = ":"))
    s1 <- str_trim(s_parts[1])
    s2 <- str_trim(s_parts[2])
    if(!is.na(s1) & !is.na(str_match(s1,"[0-9]{2,3}")) & s1==str_match(s1,"[0-9]{2,3}") &
       !is.na(s2) & !is.na(str_match(s2,"[0-9]{2,3}")) &  s2==str_match(s2,"[0-9]{2,3}")) {
      result <- paste0(s1,":",s2)
    }
  }
  return(result)
}

#Target raw data filename for later use
rawdata_file <- "imgt_cell_dat_raw.csv"

#Only download if not exists, download can take hours...
if(!file.exists(rawdata_file)) {
  # Get Cell IDs :
  i<-1
  error_code <- 0
  baseurl<-"https://www.ebi.ac.uk/cgi-bin/ipd/api/cell"
  nexturl<-"?limit=1000"
  id_list <- c()
  while (error_code==0) {
    print(paste0("Downloading index list ",i))
    get_idlist_json <- fromJSON(paste0(baseurl,nexturl), flatten = TRUE)
    if(length(get_idlist_json$data)==2) {
      id_list <- c(id_list,get_idlist_json$data$cellid)
      if(!is.null(get_idlist_json$meta$`next`)) {
        nexturl <- get_idlist_json$meta$`next`
      }else {
        error_code <- -1
      }
    }else {
      error_code <- -2
    }
    i <- i+1
  }
  length(id_list)

  # Download cell data for each cell ID :
  rawdat <- map_df(1:length(id_list), function(i) {
    id <- id_list[i]
    print(paste0("Downloading cell data ",i,"/",length(id_list)))
    cell_url <- paste0("https://www.ebi.ac.uk/cgi-bin/ipd/api/cell/",id)
    cell_data <- fromJSON(cell_url, flatten = TRUE)
    i<-i+1
    data.frame(cellid = cell_data$cellid,
               ancestry = ConvertNull(cell_data$source$ancestry),
               HLA_A_1 = ConvertNull(cell_data$typing$A[1]),
```

```
                HLA_A_2 = ConvertNull(cell_data$typing$A[2]),
                HLA_B_1 = ConvertNull(cell_data$typing$B[1]),
                HLA_B_2 = ConvertNull(cell_data$typing$B[2]),
                HLA_C_1 = ConvertNull(cell_data$typing$C[1]),
                HLA_C_2 = ConvertNull(cell_data$typing$C[2]))
  })

  #Save result file for later use
  write.csv(rawdat,rawdata_file,row.names=FALSE)

  #Remove variables
  rm(id_list)
  rm(rawdat)
}

#Reload the (pre) downloaded rawdata file
rawdat <- read.csv(rawdata_file,header=TRUE)
```

Data cleaning

Data cleaning is required to convert the allele names and descents into a useful format and filter out subjects with insufficient data.

Convert the alleles to 4 digit format by using the predefined helper function.

```
#Convert to 4 digit allele format
rawdat[,3:8] <- apply(rawdat[,3:8],c(1,2),convert_4_digit_allele)
```

In the next step, we filter the cells to have the 6 alleles and convert ancestry texts to descents. Only samples (cells) with valid descent are kept.

```
#Generate cleared data (all allele is fine, descent is valid):
data_cleared <- rawdat %>%
  filter(!is.na(HLA_A_1) & !is.na(HLA_A_2) &
           !is.na(HLA_B_1) & !is.na(HLA_B_2) &
           !is.na(HLA_C_1) & !is.na(HLA_C_2)) %>%
  mutate(Descent = NA,
         Descent = ifelse(str_starts(ancestry,"Admixed"),"Admixed",Descent),
         Descent = ifelse(str_starts(ancestry,"African"),"African",Descent),
         Descent = ifelse(str_starts(ancestry,"Asian"),"Asian",Descent),
         Descent = ifelse(str_starts(ancestry,"European"),"European",Descent),
         Descent = ifelse(str_starts(ancestry,"Greater Middle Eastern"),"Greater Middle Eastern",Descen
         Descent = ifelse(str_starts(ancestry,"Hispanic or Latin American"),"Hispanic or Latin American
         Descent = ifelse(str_starts(ancestry,"Native American"),"Native American",Descent),
         Descent = ifelse(str_starts(ancestry,"Oceanian"),"Oceanian",Descent),
         Descent = ifelse(str_starts(ancestry,"Aboriginal Australian"),"Aboriginal Australian",Descent)
         Descent = ifelse(str_starts(ancestry,"Undefined"),"Undefined",Descent)) %>%
  filter(!is.na(Descent)) %>%
  select(cellid,Descent,HLA_A_1,HLA_A_2,HLA_B_1,HLA_B_2,HLA_C_1,HLA_C_2)
```

In the next step we extend the allele digits with the proper allele name. This will be very useful when we convert the data to long format and the numeric 4 digit allele format is not enough to identify the allele type. For example 02:01 from the HLA-A column will be renamed to HLA-A*02:01.

```r
#Add allele prefix before the HLA digits
data_cleared$HLA_A_1=paste0("HLA-A*",data_cleared$HLA_A_1)
data_cleared$HLA_A_2=paste0("HLA-A*",data_cleared$HLA_A_2)
data_cleared$HLA_B_1=paste0("HLA-B*",data_cleared$HLA_B_1)
data_cleared$HLA_B_2=paste0("HLA-B*",data_cleared$HLA_B_2)
data_cleared$HLA_C_1=paste0("HLA-C*",data_cleared$HLA_C_1)
data_cleared$HLA_C_2=paste0("HLA-C*",data_cleared$HLA_C_2)
```

Data exploration

Our data has 32344 rows and the following data structure:

cellid	Descent	HLA_A_1	HLA_A_2	HLA_B_1	HLA_B_2	HLA_C_1	HLA_C_2
10007	European	HLA-A*01:01	HLA-A*24:02	HLA-B*08:20	HLA-B*35:03	HLA-C*07:01	HLA-C*12:03
10022	African	HLA-A*02:01	HLA-A*68:01	HLA-B*45:01	HLA-B*58:02	HLA-C*06:02	HLA-C*16:01
10033	African	HLA-A*11:01	HLA-A*30:01	HLA-B*53:01	HLA-B*55:01	HLA-C*03:03	HLA-C*04:13
10035	African	HLA-A*32:01	HLA-A*33:03	HLA-B*14:01	HLA-B*27:03	HLA-C*03:04	HLA-C*08:02
10066	European	HLA-A*01:01	HLA-A*02:24	HLA-B*07:02	HLA-B*37:01	HLA-C*06:02	HLA-C*07:02
10075	African	HLA-A*30:01	HLA-A*68:01	HLA-B*42:01	HLA-B*45:01	HLA-C*16:01	HLA-C*17:01

The descent distribution is like this :

Descent	N
Undefined	25233
European	2839
Asian	2548
African	793
Hispanic or Latin American	386
Admixed	366

Unfortunately we have a lot of unknown descents. We will focus on the European and Asian ones because they have around the similar counts.

```r
dat <- data_cleared %>%
  filter(Descent %in% c("Asian","European"))
nrow(dat)
```

```
## [1] 5387
```

After data cleaning we have 5387 rows with around half Asian and half European descent. We will make a 10 percent test set and a 90 percent train set to develop our models. At the end we will use 10-fold cross validation to get the average accuracy without having an independent validation set. I use this strategy because the limited number of data.

```r
#Before we continue, split the data into train and test set in the ratio of 90% vs 10%
set.seed(1)
test_index = createDataPartition(dat$Descent, times = 1, p = 0.1, list = FALSE)
train_set <- dat[-test_index,]
test_set <- dat[test_index,]
```

The test set has 539 and the train set has 4848 observations. From now we will use the train set to characterize our model.

To check the frequency of alleles we will transform the data into long format and count the alleles.

```r
#Create long format of the train set:
train_set_long <- train_set %>%
  pivot_longer(cols=-c("cellid","Descent"),names_to="HLAGroup",values_to="HLAValue")

#Check HLA frequency (use distinct because homozigotes)
hla_freq <- train_set_long %>%
  select(cellid,HLAValue) %>%
  distinct() %>%
  group_by(HLAValue) %>%
  summarise(N=n(),Percentage=100*N/nrow(train_set)) %>%
  arrange(desc(N))
knitr::kable(head(hla_freq),row.names=FALSE)
```

HLAValue	N	Percentage
HLA-A*02:01	1646	33.95215
HLA-A*24:02	1261	26.01073
HLA-C*07:02	1250	25.78383
HLA-A*11:01	1009	20.81271
HLA-A*01:01	802	16.54290
HLA-C*04:01	747	15.40842

We can see that there are common alleles. The top 100 HLA frequency looks like this :

```r
#ggplot of the HLA frequencies of the top 100 HLA (ordered):
head(hla_freq,100) %>%
  ggplot(aes(reorder(HLAValue, -Percentage),Percentage)) +
  geom_bar(stat="identity") + xlab("TOP100 HLA alleles") +
  theme(axis.text.x=element_blank(),axis.ticks.x = element_blank())
```

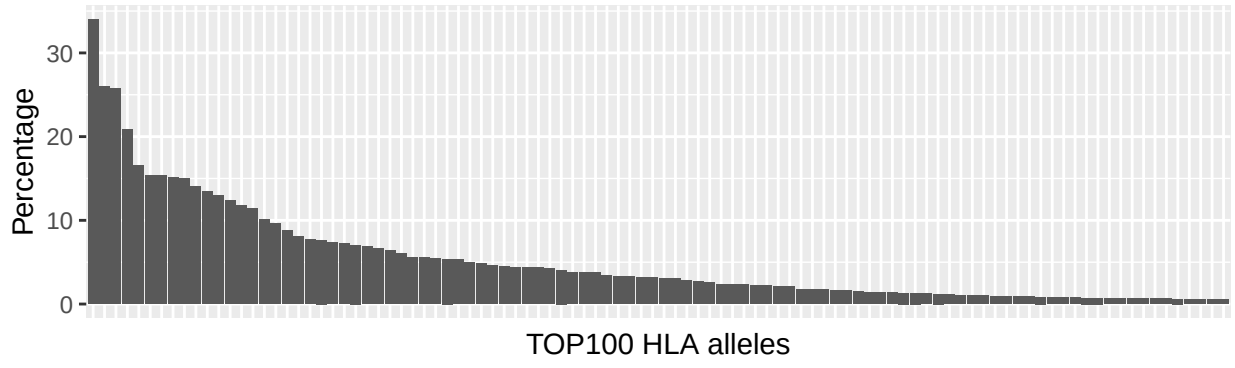


Table 4: Asian

HLAValue	N	Percentage
HLA-A*24:02	788	34.36546
HLA-A*11:01	717	31.26908
HLA-C*07:02	634	27.64937
HLA-C*01:02	558	24.33493
HLA-A*02:01	483	21.06411

Table 5: European

HLAValue	N	Percentage
HLA-A*02:01	1163	45.51859
HLA-A*01:01	655	25.63601
HLA-C*07:01	638	24.97065
HLA-A*03:01	633	24.77495
HLA-C*07:02	616	24.10959

Compare top HLA frequencies for the descents.

We see that the top frequencies show differences between the two descents. For example the top HLA for European is HLA-A∗02:01, but he same allele is only number 5 for Asian. The most frequent allele for Asian is HLA-A∗24:02 but it is not listed in the top 5 of European HLA.

Methods

Preparation with One-Hot Encoding

In the first three model we will apply the classification tree, the KNN and the GLM methods. The test set contains around 2000 HLAs what is a lot of HLA combinations for the subject's 6 MHCI allele. Converting the HLA names into numbers for the target methods method can lead to problems, because the order of the alleles may interferes the result of the algorithm. Having an HLA is basically a categorical data, so we can use One-Hot Encoding to deal with it. For this, we have to convert our data set into a much wider format, where every possible HLA has a column and the the value is 1 if that subject has the HLA and the value is 0 if the subject does not have that HLA. For practical reasons we will not convert 2000 HLA to columns, we will use only the top 100 HLA in the frequency list. This way the model is viable for much larger populations as well, when we have much more HLAs.

One-Hot Encoding for the train set:

```
#One-Hot Encoding for the top 100 HLA
train_set_ohe <- train_set_long %>%
  filter(HLAValue %in% top100_hla_list) %>%
  select(cellid,Descent,HLAValue) %>%
  distinct() %>%
  mutate(HaveHLA=1) %>%
  pivot_wider(names_from = HLAValue, values_from = HaveHLA) %>%
  select(-cellid) %>%
  mutate(Descent=factor(Descent,levels=c("European","Asian")))

#Convert NA values to zeroes
train_set_ohe[is.na(train_set_ohe)] <- 0

#Convert columns names for train compatible format
colnames(train_set_ohe) <- make.names(colnames(train_set_ohe))
```

Our data looks like this after the One-Hot encoding (showing only the first rows and columns) :

```
knitr::kable(head(train_set_ohe[,1:6]))
```

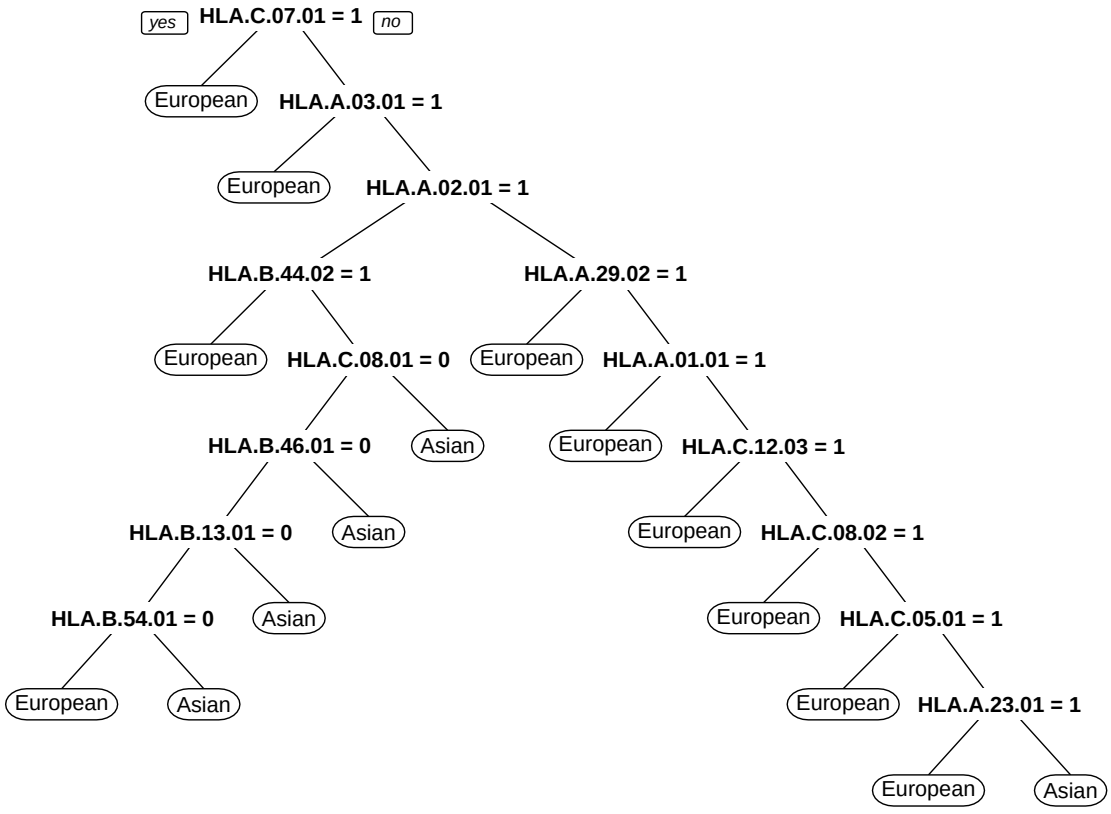
Descent	HLA.A.01.01	HLA.A.24.02	HLA.B.35.03	HLA.C.07.01	HLA.C.12.03
European	1	1	1	1	1
European	1	0	0	0	0
European	0	0	0	0	0
Asian	0	0	0	0	0
European	0	0	0	0	0
European	0	0	0	1	0

We do the same transformation for the test set. The code is not included here, but can be checked in the Rmd or R source.

Now we are ready to apply some traditional models :

Classification tree

```
#Classification tree :
fit_class <- rpart(Descent ~ ., data = train_set_ohe)
rpart.plot::prp(fit_class,uniform = TRUE, compress = TRUE, branch = .2)
```



```r
#Classification tree performance on the test set
y_hat_class <- predict(fit_class, test_set_ohe, type = "class")
mean(y_hat_class==test_set_ohe$Descent)
```

```
## [1] 0.8240512
```

The accuracy is not bad. We can see some alleles from the TOP5 lists because the common alleles play a role in determining descents. The top level allele is HLA-C*07:01, this means that if the subject has this allele we predict European. Let's check the the ratios for this allele for the test set :

Descent	N	Percentage
Asian	42	1.831662
European	638	24.970646

There is a big difference between the 2 ratios. This allele is indeed more possible for Europeans.

KNN model

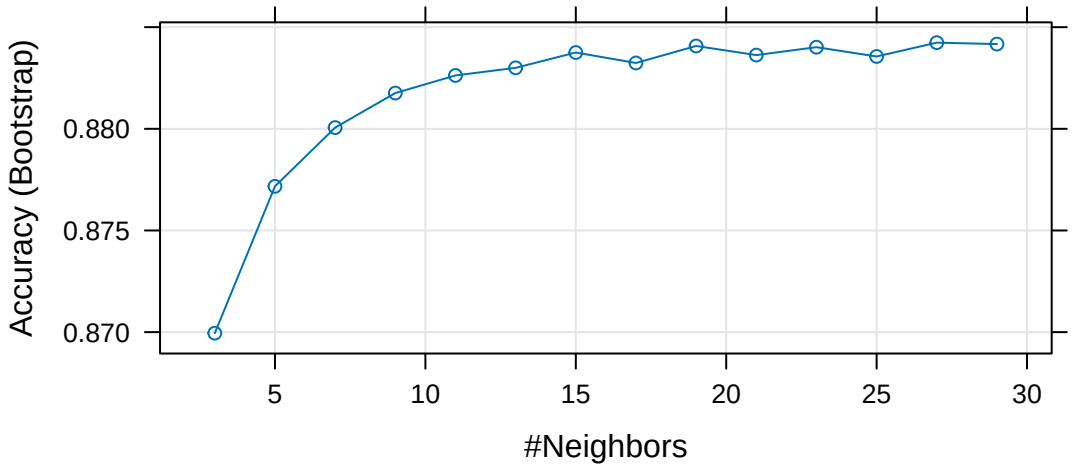
For the KNN method we will try different k values from 3 to 29.

```r
#Knn model:
knn_fit <- train(Descent ~ ., method = "knn", data = train_set_ohe,
                 tuneGrid = data.frame(k = seq(3,29,by = 2)))
knn_fit$bestTune
```

```
##     k
## 13 27
```

Tuning parameters:

```r
plot(knn_fit)
```



Accuracy on the test set :

```
y_hat_knn <- predict(knn_fit, test_set_ohe, type = "raw")
mean(y_hat_knn==test_set_ohe$Descent)
```

```
## [1] 0.8921205
```

The KNN method has better accuracy. The optimal k parameter is surprisingly large, but the difference between the k values from 15 to 31 is not too high.

GLM Model

```
glm_fit <- train(Descent ~ ., method = "glm", data = train_set_ohe)
y_hat_glm <- predict(glm_fit, test_set_ohe, type = "raw")
mean(y_hat_glm==test_set_ohe$Descent)
```

```
## [1] 0.9115099
```

The GLM method performs quite well. This method and the previous ones can be further optimized by changing the top 100 HLA to a lower/higher amount. Instead of this let's try to develop a simple model which using all the HLA alleles.

Relative Frequency model

In this part we create a new model. For this model we will calculate relative frequencies for all HLA. Relative frequency is defined to every allele independently from others as the possibility to being Asian or being European descent.

```
#Calculate percentages of being EU or AS for each HLA
hla_freq_descent <- train_set_long %>%
  distinct() %>%
  mutate(EU = ifelse(Descent=="European",1,0),
         AS = ifelse(Descent=="Asian",1,0)) %>%
  group_by(HLAValue) %>%
  summarize(N=n(),EU=sum(EU,na.rm = TRUE)/N,AS=sum(AS,na.rm = TRUE)/N) %>%
  arrange(desc(N))
knitr::kable(head(hla_freq_descent))
```

HLAValue	N	EU	AS
HLA-A*02:01	1777	0.7175014	0.2824986
HLA-A*24:02	1337	0.3649963	0.6350037
HLA-C*07:02	1312	0.4923780	0.5076220
HLA-A*11:01	1093	0.2689844	0.7310156
HLA-A*01:01	841	0.8204518	0.1795482
HLA-C*04:01	779	0.6829268	0.3170732

This means if we pick up HLA-A*02:01 that it is European descent in 72% and Asian in 28%. If we pick up HLA-A*24:02, it is Asian descent in 64% and European in 36%. And so on for the other alleles. By using this informations we can can build and algorith to summarize this values for the subjects.

Method performance on the test set:

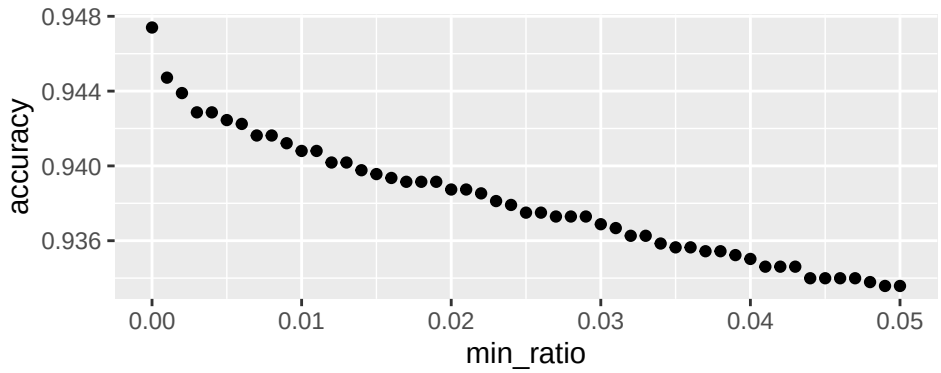
```
test_set_long %>%
  left_join(hla_freq_descent,by=c("HLAValue"))  %>%
  group_by(cellid,Descent) %>%
  summarize(EU=prod(EU,na.rm = TRUE),AS=prod(AS,na.rm = TRUE)) %>%
  mutate(predicted_descent = ifelse(EU>AS,"European","Asian"),
         ok=ifelse(predicted_descent==Descent,1,0)) %>%
  ungroup() %>%
  summarize(Method="Frequency model 1 (test set)",accuracy=sum(ok,na.rm = TRUE)/n()) %>%
  knitr::kable()
```

Method	accuracy
Frequency model 1 (test set)	0.947401

Optimization for minimal ratio because zero and 100% values. Let's try to increase 0% values and decrease 100% values with a constant.

```
result_optimization <- map_df(seq(0,0.05,by=0.001),function(min_ratio) {
  test_set_long %>%
    left_join(hla_freq_descent,by=c("HLAValue"))  %>%
    mutate(EU=pmax(EU,min_ratio),AS=pmax(AS,min_ratio)) %>%
    mutate(EU=pmin(EU,1-min_ratio),AS=pmin(AS,1-min_ratio)) %>%
    group_by(cellid,Descent) %>%
    summarize(EU=prod(EU,na.rm = TRUE),AS=prod(AS,na.rm = TRUE)) %>%
    mutate(predicted_descent = ifelse(EU>AS,"European","Asian"),
           ok=ifelse(predicted_descent==Descent,1,0)) %>%
    ungroup() %>%
    summarize(N=n(),min_ratio=min_ratio,accuracy=sum(ok,na.rm = TRUE)/N)
})
```

```
result_optimization %>%
  ggplot(aes(min_ratio,accuracy)) +
  geom_point()
```



As we see the the optimization does not really help to improve our previous accuracy. So we stay with the previous version.

This result is based on a relatively small test group, so better to do a 10-fold cross validation to get the real accuracy. We do this only for our selected method.

10-fold Cross validation

For cross-validation, the entire data set is divided into 10 parts. We perform the validation process on each part, as this is the test set and the rest of the dataset is the training set. At the end, we calculate the average accuracy.

```
#Manually create 10 fold for the cross validation:
set.seed(1)
split_index <- createFolds(dat$Descent, k = 10, list = FALSE, returnTrain = FALSE)

#Cross validation for each fold:
cv_result <- map_df(seq(1,10,1), function(fold_index) {
  index_test <- which(split_index==fold_index)
  train_set <- dat[-index_test,]
  test_set <- dat[index_test,]
  #Create long format of the train set and determine HLA frequencies:
  hla_freq_descent <- train_set %>%
    pivot_longer(cols=-c("cellid","Descent"),names_to="HLAGroup",values_to="HLAValue") %>%
    distinct() %>%
    mutate(EU = ifelse(Descent=="European",1,0),
           AS = ifelse(Descent=="Asian",1,0)) %>%
    group_by(HLAValue) %>%
    summarize(N=n(),EU=sum(EU,na.rm = TRUE)/N,AS=sum(AS,na.rm = TRUE)/N)
  #Calculate accuracy
  test_set %>%
    pivot_longer(cols=-c("cellid","Descent"),names_to="HLAGroup",values_to="HLAValue") %>%
    left_join(hla_freq_descent,by=c("HLAValue"))  %>%
    group_by(cellid,Descent) %>%
    summarize(EU=prod(EU,na.rm = TRUE),AS=prod(AS,na.rm = TRUE)) %>%
    mutate(predicted_descent = ifelse(EU>AS,"European","Asian"),
           ok=ifelse(predicted_descent==Descent,1,0)) %>%
    ungroup() %>%
    summarize(Method="Frequency model",fold_index=fold_index,accuracy=sum(ok,na.rm = TRUE)/n())
})

#Calculate accuracy mean:
real_accuracy <- mean(cv_result$accuracy)
real_accuracy
```

```
## [1] 0.9053241
```

Results

At the end we have 0.91 cross validated accuracy, this match our goal to have at least 90% for this two descents. The developed Relative Frequency model is fast, scalable for subject size and pretty accurate. This result shows that there are real differences in the MHCI HLA set of the Asian and European populations.

The finding is not really new, but the project was a great opportunity for me to study the IPD-IMGT/HLA database and improve my R skills.

Conclusions

Presumably the model performance will drop if we try to add more descents. For example adding descents which are naturally lives close to each other will introduce more similarities in HLA sets and making harder to distinguish them. We could also improve the method to use a different calculation for homozygous alleles, when a subject have inherited the same alleles from parent (like having two HLA-A∗02:01). I believe that population-based or HLA-tailored vaccine designs may have a place in vaccine development.

References

The data is downloaded from :
IPD-IMGT/HLA database
https://www.ebi.ac.uk/ipd/imgt/hla/
https://www.ebi.ac.uk/ipd/imgt/hla/licence/
Barker DJ, Maccari G, Georgiou X, Cooper MA, Flicek P, Robinson J, Marsh SGE
The IPD-IMGT/HLA Database
Nucleic Acids Research (2023) 51:D1053-60

You can read more about MHC and HLA here:
Major histocompatibility complex Wikipedia page
https://en.wikipedia.org/wiki/Major_histocompatibility_complex