```r
#install and load needed libraries
libraries <- c("tidyverse", "ggpubr", "factoextra", "randomForest", "vcfR",
"caret", "rfviz", "clValid", "ClusterR")
install.packages(setdiff(libraries, rownames(installed.packages())))
lapply(libraries, library, character.only = TRUE)

#set seed
set.seed(500)

#create function that will filter any raw vcf file retaining only the
genotype data
filter_vcf <- function(raw_vcf){
  vcf <- extract.indels(raw_vcf, return.indels = FALSE)
  vcf <- vcf@gt
  vcf <- vcf[ , -1]
  return(vcf)
}

#import the sub population names and codes for each mega population
igsr_pop <- read_tsv("igsr_populations.tsv")

## Rows: 212 Columns: 11
## -- Column specification --------------------------------------------------
------
## Delimiter: "\t"
## chr (8): Population code, Population elastic ID, Population name,
Population...
## dbl (3): Population latitude, Population longitude, Superpopulation
display ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.

colnames(igsr_pop) <- gsub(" ", "_", colnames(igsr_pop))

#save the subpopulation codes for eu and eas
igsr_eu <- igsr_pop %>%
  filter(Superpopulation_code == "EUR")
igsr_eas <- igsr_pop %>%
  filter(Superpopulation_code == "EAS")

#save the subpopulation codes for each mega population
subpop_eu <- unique(igsr_eu$Population_code)
subpop_eas <- unique(igsr_eas$Population_code)

#read the rwa vcf files
betaglob_eu <- read.vcfR("betaglob_eu.vcf")
```

```
## Scanning file to determine attributes.
## File attributes:
##   meta lines: 253
##   header_line: 254
##   variant count: 136
##   column count: 512
## Meta line 253 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 136
##   Character matrix gt cols: 512
##   skip: 0
##   nrows: 136
##   row_num: 0
## Processed variant: 136
## All variants processed
```

```
betaglob_eas <- read.vcfR("betaglob_eas.vcf")
```

```
## Scanning file to determine attributes.
## File attributes:
##   meta lines: 253
##   header_line: 254
##   variant count: 136
##   column count: 513
## Meta line 253 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 136
##   Character matrix gt cols: 513
##   skip: 0
##   nrows: 136
##   row_num: 0
## Processed variant: 136
## All variants processed
```

```
pdha1_eu <- read.vcfR("pdha1_eu.vcf")
```

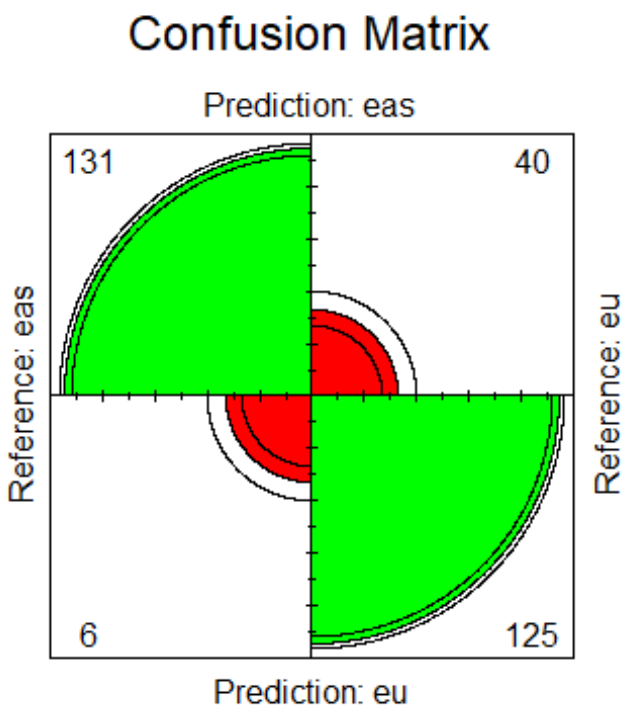
```
## Scanning file to determine attributes.
## File attributes:
##   meta lines: 254
##   header_line: 255
##   variant count: 401
##   column count: 512
## Meta line 254 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 401
```

```
##   Character matrix gt cols: 512
##   skip: 0
##   nrows: 401
##   row_num: 0
## Processed variant: 401
## All variants processed

pdha1_eas <- read.vcfR("pdha1_eas.vcf")

## Scanning file to determine attributes.
## File attributes:
##   meta lines: 254
##   header_line: 255
##   variant count: 401
##   column count: 513
## Meta line 254 read in.
## All meta lines processed.
## gt matrix initialized.
## Character matrix gt created.
##   Character matrix gt rows: 401
##   Character matrix gt cols: 513
##   skip: 0
##   nrows: 401
##   row_num: 0
## Processed variant: 401
## All variants processed

#filter the vcf files retaining only the genotype matrix
betaglob_eu_gt <- filter_vcf(betaglob_eu)
betaglob_eas_gt <- filter_vcf(betaglob_eas)
pdha1_eu_gt <- filter_vcf(pdha1_eu)
pdha1_eas_gt <- filter_vcf(pdha1_eas)

#bind both genes from each megapopulation to each other
eu_gt <- rbind(betaglob_eu_gt, pdha1_eu_gt)
eas_gt <- rbind(betaglob_eas_gt, pdha1_eas_gt)

#create a vector to mark samples as either eu or eas
ethn <- c(rep("eu", ncol(eu_gt)), rep("eas", ncol(eas_gt)))

#create a raw random forest dataframe binding both eu and eas samples
rf_df <- cbind(eu_gt, eas_gt)

#remove unneeded variables
rm(igsr_pop, igsr_eu, igsr_eas, subpop_eu, subpop_eas, betaglob_eas,
betaglob_eas_gt, betaglob_eu, betaglob_eu_gt, pdha1_eas, pdha1_eas_gt,
pdha1_eu, pdha1_eu_gt, eu_gt, eas_gt, libraries )

#code the genotype for homozygous dominant, heterozygous and homozygous
recessive as 0,1,2
```

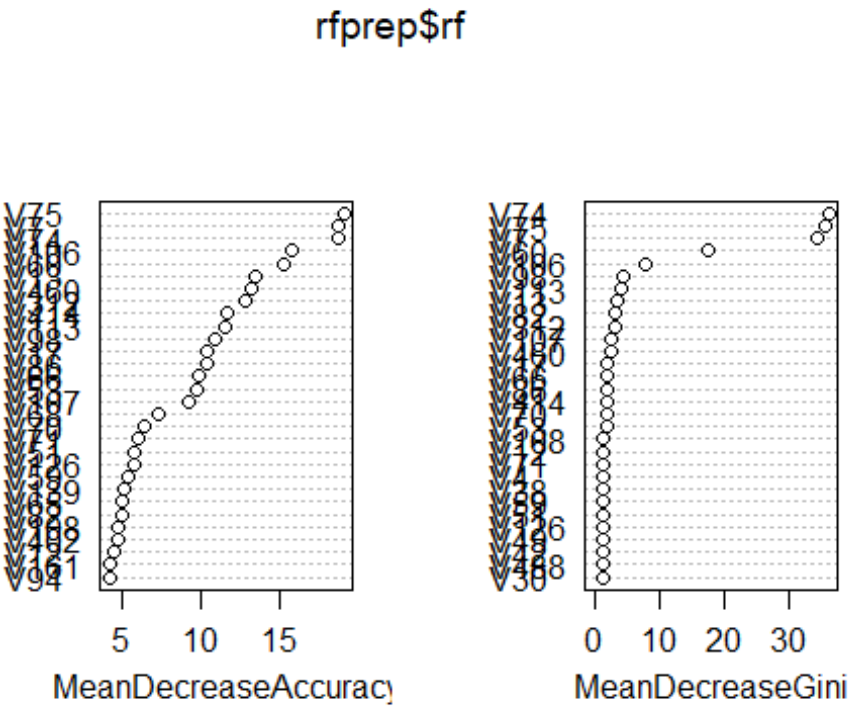
```r
rf_df[rf_df == "0|0"] <- 0
rf_df[rf_df == "0|1" | rf_df == "1|0"] <- 1
rf_df[rf_df == "1|1"] <- 2

#transpose the random forest dataframe and add the ethnicity marker as a
column
rf_df <- as.data.frame(t(rf_df)) %>%
  tibble::rownames_to_column() %>%
  add_column(ethnicity = ethn, .after = c(1))

#convert columns three and onwards to numeric for subsequent operations
rf_df[ , 3:ncol(rf_df)] <- apply(rf_df[ , 3:ncol(rf_df)], 2, function(x)
as.numeric(as.character(x)))

#scramble the order of the rows in the random forest dataframe
rf_df <- rf_df[sample(nrow(rf_df)), ]

#filter all columns in which the variance is constant (i.e columns that have
constant values throughout and therefore won't be useful for model training)
rf_df_filtered <- rf_df[ - as.numeric(which(apply(rf_df, 2, var) == 0))] %>%
  na.omit()

## Warning in FUN(newX[, i], ...): NAs introduced by coercion

## Warning in FUN(newX[, i], ...): NAs introduced by coercion

#sample 70% of the random forest dataframe for training
rf_train <- rf_df_filtered %>%
  sample_frac(0.7)

#sample the remained of the random forest dataframe for testing
rf_test <- anti_join(rf_df_filtered, rf_train, by = "rowname")

#train the random forest model using the training dataframe
ran_for <- randomForest(x = rf_train[ , 3:ncol(rf_train)], y =
as.factor(rf_train$ethnicity), data = rf_train)

#use the trained model to predict the ethnicity of the samples in the testing
dataframe
pred_randfor <- predict(ran_for, rf_test[ , 3:ncol(rf_test)])

#create a confusion matrix to show the results
cf <- confusionMatrix(pred_randfor, as.factor(rf_test$ethnicity))
cf

## Confusion Matrix and Statistics
##
##           Reference
## Prediction eas  eu
##        eas 131  40
```

```
##         eu    6 125
##
##                Accuracy : 0.8477
##                  95% CI : (0.8021, 0.8863)
##     No Information Rate : 0.5464
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6991
##
##  Mcnemar's Test P-Value : 1.141e-06
##
##             Sensitivity : 0.9562
##             Specificity : 0.7576
##          Pos Pred Value : 0.7661
##          Neg Pred Value : 0.9542
##              Prevalence : 0.4536
##          Detection Rate : 0.4338
##    Detection Prevalence : 0.5662
##       Balanced Accuracy : 0.8569
##
##        'Positive' Class : eas
##
```

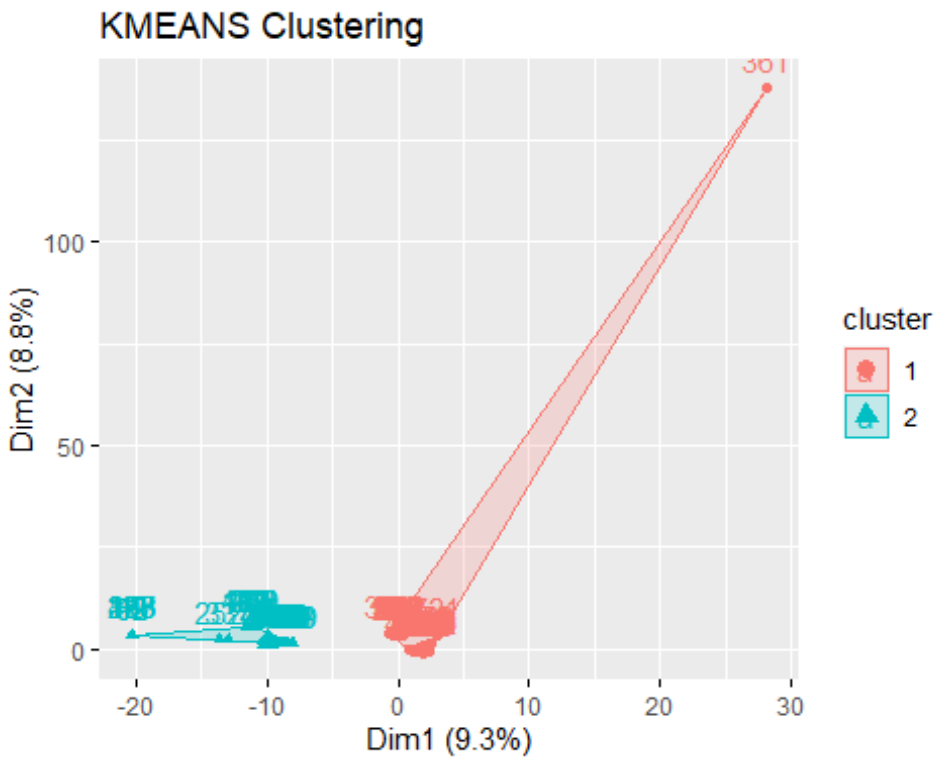
```
#plot the confusion matrix
fourfoldplot(as.table(cf), color = c("red", "green"), main = "Confusion
Matrix")
```



```r
#plot the predictors against their impact on the mean decrease to show which predictors are most important in training the model
rfprep <- rf_prep(x = rf_train[ , 3:ncol(rf_train)], y = as.factor(rf_train$ethnicity), data = rf_train)
varImpPlot(rfprep$rf)
```

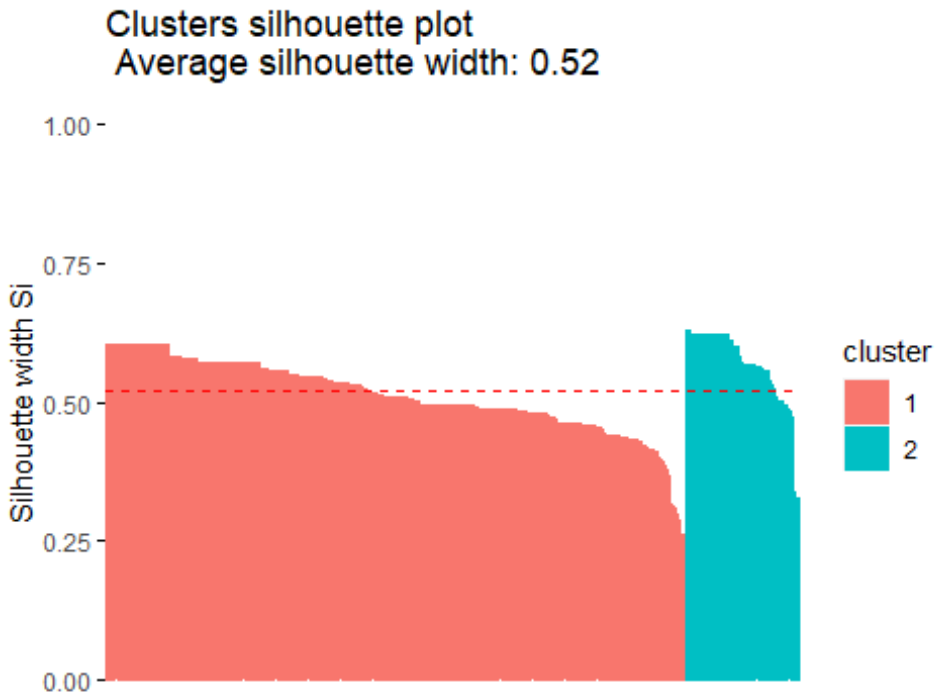



```r
#use the kmeans algorithm with two clusters to group the samples and plot the clusters
kmeans <- eclust(rf_df_filtered[ , 3:ncol(rf_df_filtered)], "kmeans", nstart = 30, k = 2)
```




```
#plot the silhouette width of each cluster alongside the silhouette value
fviz_silhouette(kmeans)
```

```
##   cluster size ave.sil.width
## 1       1  843          0.51
## 2       2  164          0.56
```




```
#print the dunn_index value
dunn(clusters = kmeans$cluster, Data = rf_df_filtered[ ,
3:ncol(rf_df_filtered)])
```

```
## [1] 0.6807456
```

```
#create a validation dataframe for external validation of the clustering
algorithm
validate_df <- rf_df_filtered %>%
  select(ethnicity) %>%
  add_column(validate = 0)
validate_df[validate_df$ethnicity == "eas", 2] <- 1

#external validation of the clustering algorithm
external_validation(clusters = kmeans$cluster, true_labels =
validate_df$validate, summary_stats = TRUE)
```

```
##
## ---------------------------------------
## purity                          : 0.5998
## entropy                         : 0.5855
## normalized mutual information   : 0.0677
## variation of information        : 1.5299
## normalized var. of information : 0.9649
## ---------------------------------------
## specificity                     : 0.2926
## sensitivity                     : 0.7467
```

```
## precision                      : 0.513
## recall                         : 0.7467
## F-measure                      : 0.6082
## ---------------------------------------
## accuracy OR rand-index         : 0.5194
## adjusted-rand-index            : 0.0393
## jaccard-index                  : 0.437
## fowlkes-mallows-index          : 0.619
## mirkin-metric                  : 486824
## ---------------------------------------

## [1] 0.03932006

#remove unneeded variables
rm(list = ls())
```