This is the same dataset as phase2

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.1
## ✓ purrr     1.0.4
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at
https://goo.gl/ve3WBa
```

```
library(ggplot2)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.19.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at:
https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:
suppressPackageStartupMessages(library(dendextend))
## ---------------------
##
##
## Attaching package: 'dendextend'
##
## The following object is masked from 'package:stats':
##
##     cutree
```

```r
library(FactoMineR)
library(ggcorrplot)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(rpart)
```

```
##
## Attaching package: 'rpart'
##
## The following object is masked from 'package:dendextend':
##
##     prune
```

```r
library(rpart.plot)
library(caret)
library(caTools)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.2
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
df <- read_csv(data2)
```

```
## Rows: 4412 Columns: 11
## ── Column specification
──────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): SEX, SOURCE
## dbl (9): HAEMATOCRIT, HAEMOGLOBINS, ERYTHROCYTE, LEUCOCYTE, THROMBOCYTE,
MCH...
##
## 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.
```

```
df$SOURCE <- as.factor(df$SOURCE)
df$SEX <- as.factor(df$SEX)
dim(df)
```

```
## [1] 4412   11
```

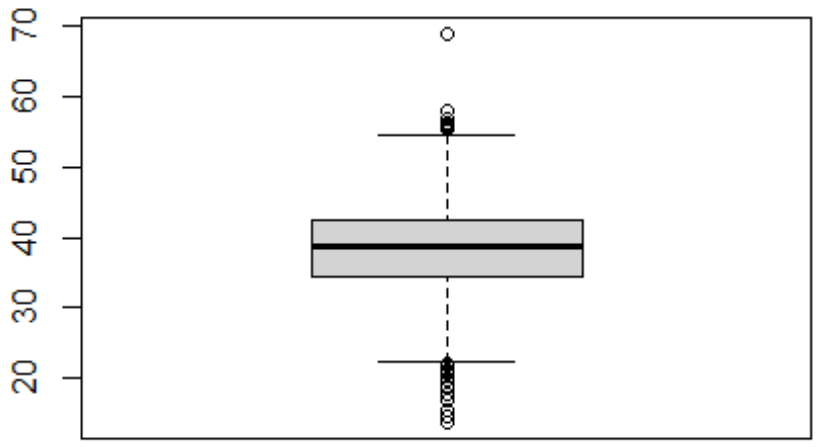
```
str(df)
```

```
## spc_tbl_ [4,412 × 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ HAEMATOCRIT : num [1:4412] 35.1 43.5 33.5 39.1 30.9 34.3 31.1 40.3 33.6
35.4 ...
##  $ HAEMOGLOBINS: num [1:4412] 11.8 14.8 11.3 13.7 9.9 11.6 8.7 13.3 11.5
11.4 ...
##  $ ERYTHROCYTE : num [1:4412] 4.65 5.39 4.74 4.98 4.23 4.53 5.06 4.73 4.54
4.8 ...
##  $ LEUCOCYTE   : num [1:4412] 6.3 12.7 13.2 10.5 22.1 6.6 11.1 8.1 11.4
2.6 ...
##  $ THROMBOCYTE : num [1:4412] 310 334 305 366 333 185 416 257 262 183 ...
##  $ MCH         : num [1:4412] 25.4 27.5 23.8 27.5 23.4 25.6 17.2 28.1 25.3
23.8 ...
##  $ MCHC        : num [1:4412] 33.6 34 33.7 35 32 33.8 28 33 34.2 32.2 ...
##  $ MCV         : num [1:4412] 75.5 80.7 70.7 78.5 73 75.7 61.5 85.2 74
73.8 ...
##  $ AGE         : num [1:4412] 1 1 1 1 1 1 1 1 1 1 ...
##  $ SEX         : Factor w/ 2 levels "F","M": 1 1 1 1 2 2 1 1 1 1 ...
##  $ SOURCE      : Factor w/ 2 levels "in","out": 2 2 2 2 2 2 2 2 2 2 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   HAEMATOCRIT = col_double(),
##   ..   HAEMOGLOBINS = col_double(),
```

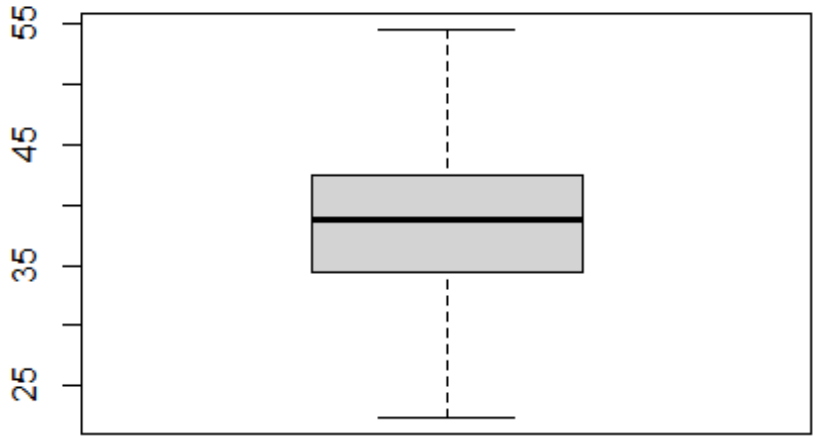
```
##   ..   ERYTHROCYTE = col_double(),
##   ..   LEUCOCYTE = col_double(),
##   ..   THROMBOCYTE = col_double(),
##   ..   MCH = col_double(),
##   ..   MCHC = col_double(),
##   ..   MCV = col_double(),
##   ..   AGE = col_double(),
##   ..   SEX = col_character(),
##   ..   SOURCE = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Removing outliers: it is very important to remove outliers before performing PCA and especially clustering. Outliers can affect the direction of the principal component loading vectors. In addition, k-means and hierarchical clustering force every observation into a cluster hence, the clusters found may be heavily distorted due to the presence of outliers that do not belong to any cluster.

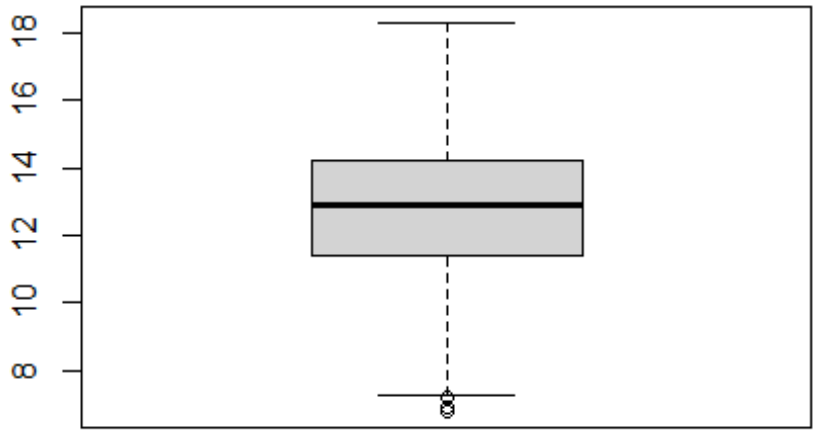
```
boxplot(df$HAEMATOCRIT)
```



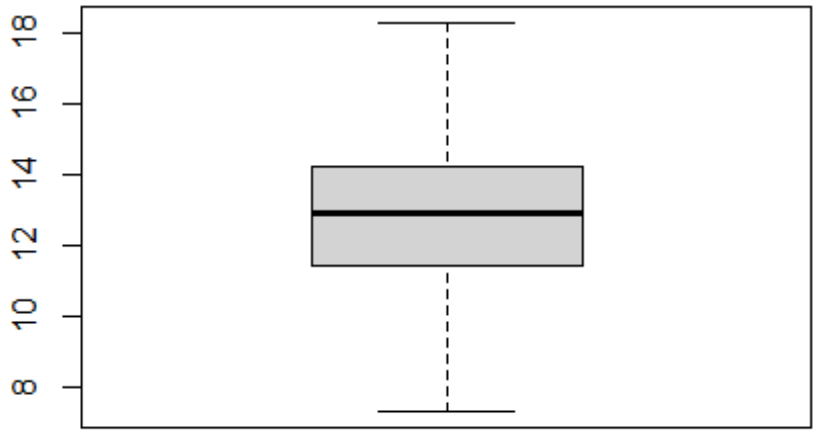

```
outliers <- boxplot(df$HAEMATOCRIT, plot=FALSE)$out
df<- df[-which(df$HAEMATOCRIT %in% outliers),]
boxplot(df$HAEMATOCRIT)
```



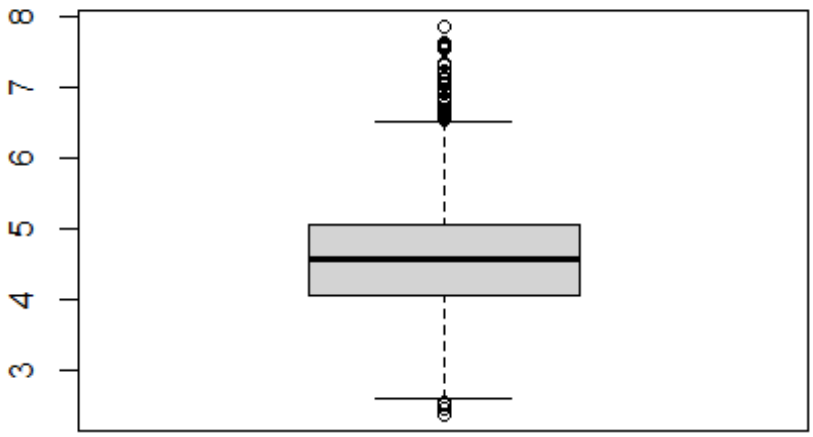

```r
boxplot(df$HAEMOGLOBINS)
```



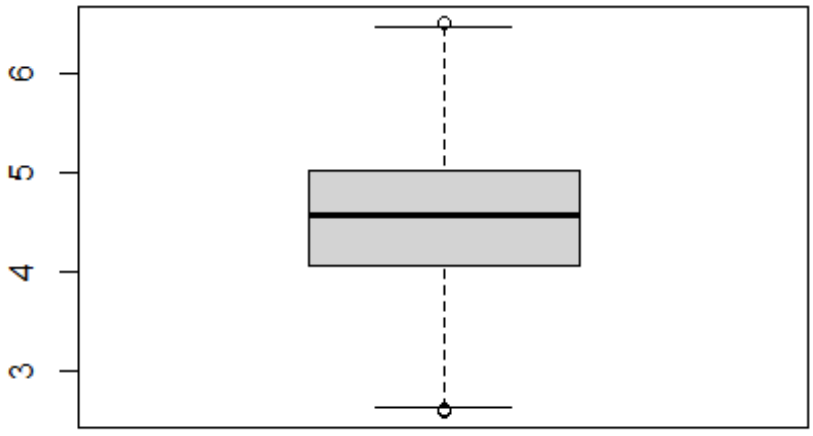
```r
outliers1 <- boxplot(df$HAEMOGLOBINS, plot=FALSE)$out
df<- df[-which(df$HAEMOGLOBINS %in% outliers1),]
boxplot(df$HAEMOGLOBINS)
```



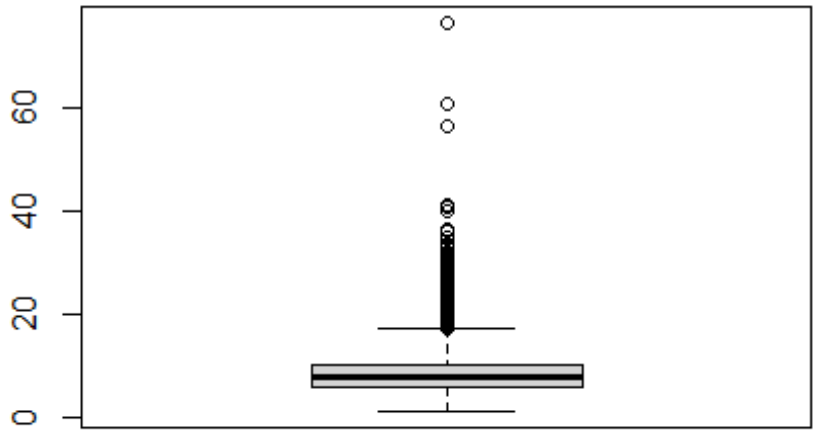

```r
boxplot(df$ERYTHROCYTE)
```



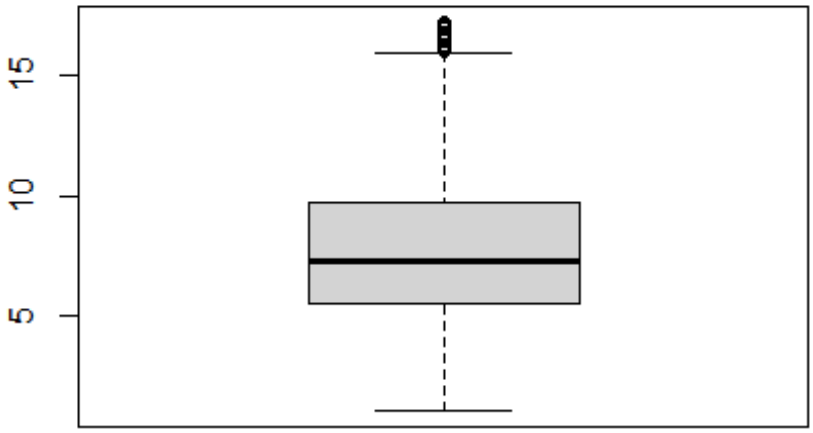

```
outliers2 <- boxplot(df$ERYTHROCYTE, plot=FALSE)$out
df<- df[-which(df$ERYTHROCYTE %in% outliers2),]
boxplot(df$ERYTHROCYTE)
```



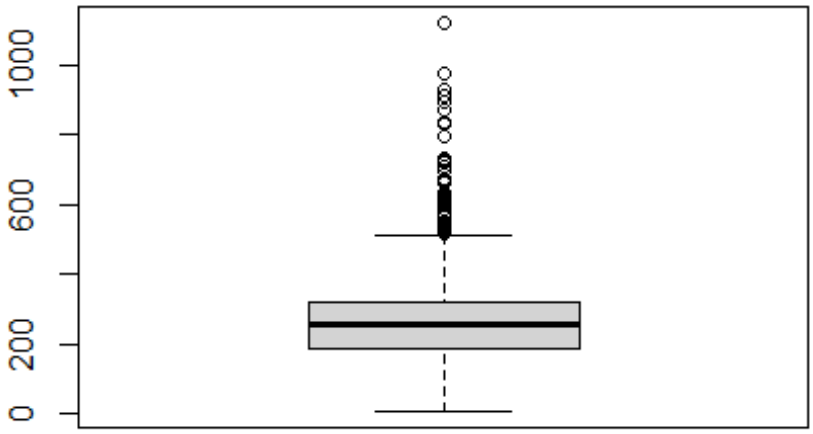
```r
boxplot(df$LEUCOCYTE)
```



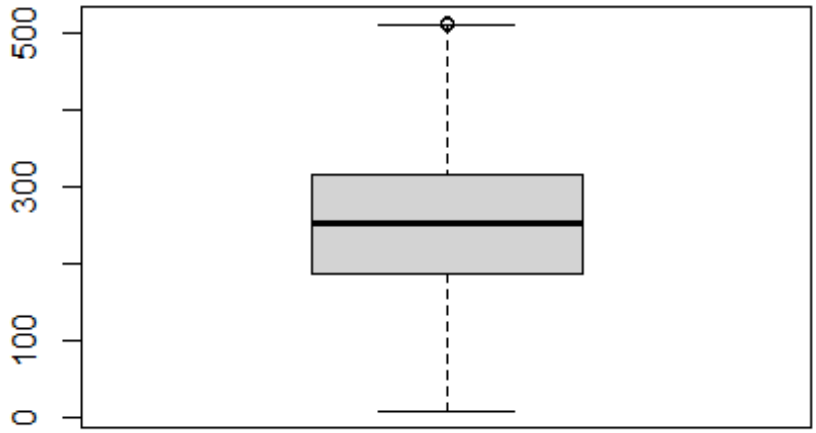

```r
outliers3 <- boxplot(df$LEUCOCYTE, plot=FALSE)$out
df<- df[-which(df$LEUCOCYTE %in% outliers3),]
boxplot(df$LEUCOCYTE)
```



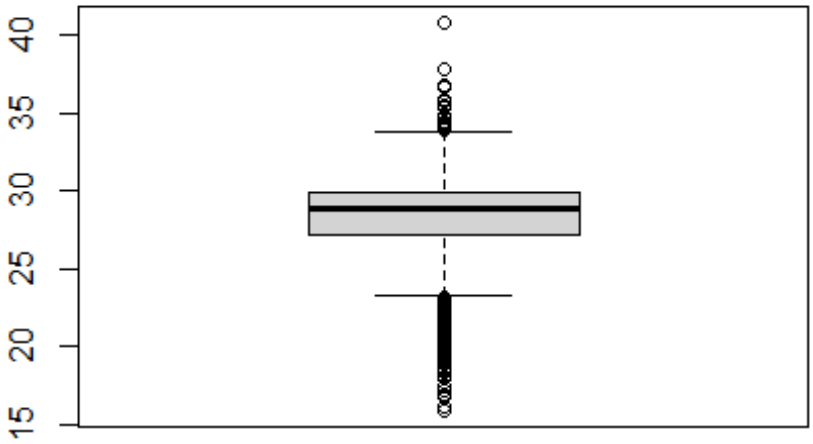

```r
boxplot(df$THROMBOCYTE)
```



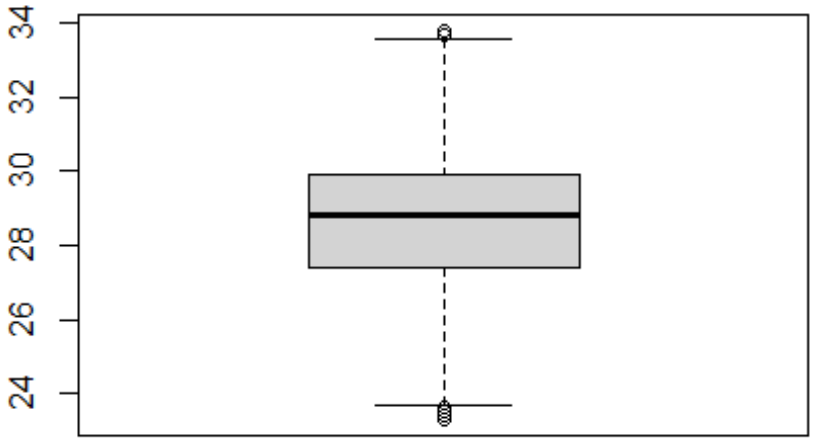
```r
outliers4 <- boxplot(df$THROMBOCYTE, plot=FALSE)$out
df<- df[-which(df$THROMBOCYTE %in% outliers4),]
boxplot(df$THROMBOCYTE)
```



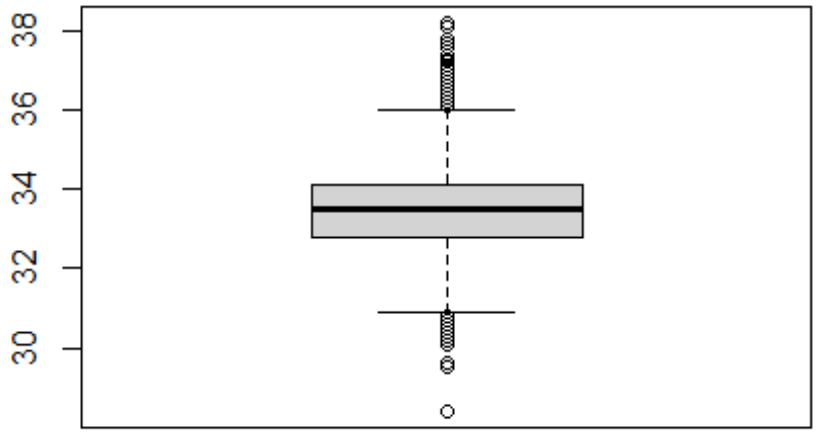

```r
boxplot(df$MCH)
```



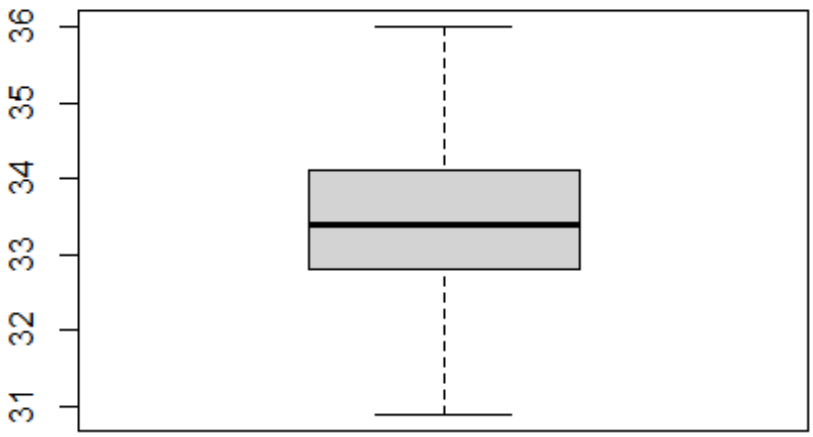

```r
outliers5 <- boxplot(df$MCH, plot=FALSE)$out
df<- df[-which(df$MCH %in% outliers5),]
boxplot(df$MCH)
```



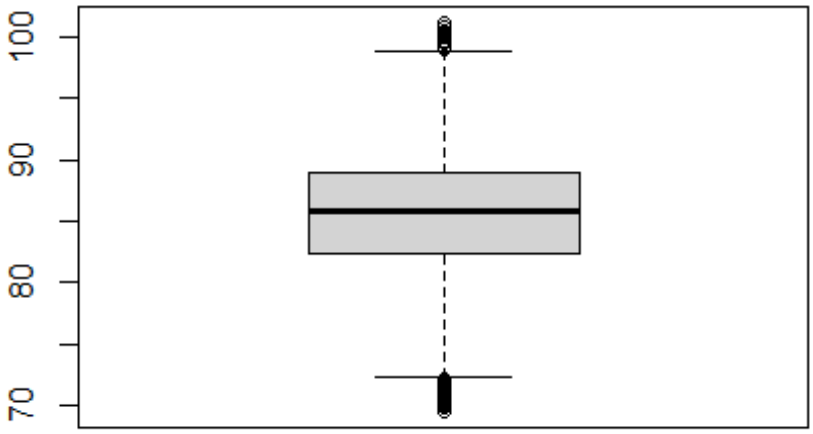
```r
boxplot(df$MCHC)
```



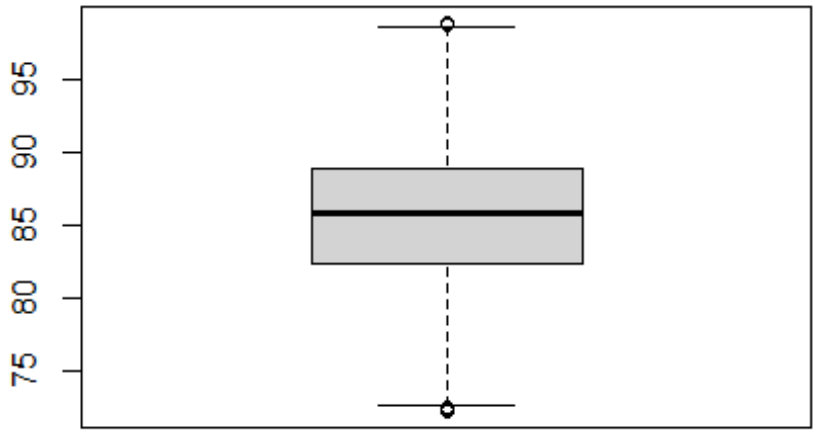

```r
outliers6 <- boxplot(df$MCHC, plot=FALSE)$out
df<- df[-which(df$MCHC %in% outliers6),]
boxplot(df$MCHC)
```




```r
boxplot(df$MCV)
```



```r
outliers7 <- boxplot(df$MCV, plot=FALSE)$out
df<- df[-which(df$MCV %in% outliers7),]
boxplot(df$MCV)
```



The number of rows decreased from 4412 observation to 3694 we removed 718 observations

```r
dim(df)
```

```
## [1] 3694   11
```

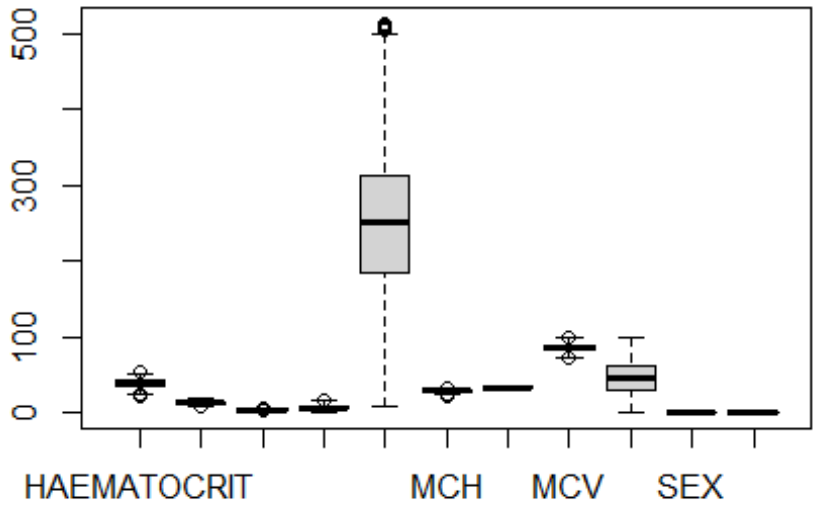
```r
str(df)
```

```
## tibble [3,694 × 11] (S3: tbl_df/tbl/data.frame)
##  $ HAEMATOCRIT : num [1:3694] 35.1 43.5 39.1 34.3 40.3 33.6 35.4 33.7 35.3
35 ...
##  $ HAEMOGLOBINS: num [1:3694] 11.8 14.8 13.7 11.6 13.3 11.5 11.4 11.5 11.9
11.6 ...
##  $ ERYTHROCYTE : num [1:3694] 4.65 5.39 4.98 4.53 4.73 4.54 4.8 4.57 4.4
4.58 ...
##  $ LEUCOCYTE   : num [1:3694] 6.3 12.7 10.5 6.6 8.1 11.4 2.6 13.2 5.8 7.4
...
##  $ THROMBOCYTE : num [1:3694] 310 334 366 185 257 262 183 322 205 154 ...
##  $ MCH         : num [1:3694] 25.4 27.5 27.5 25.6 28.1 25.3 23.8 25.2 27
25.3 ...
##  $ MCHC        : num [1:3694] 33.6 34 35 33.8 33 34.2 32.2 34.1 33.7 33.1
...
```

```
##  $ MCV          : num [1:3694] 75.5 80.7 78.5 75.7 85.2 74 73.8 73.7 80.2
76.4 ...
##  $ AGE          : num [1:3694] 1 1 1 1 1 1 1 1 1 1 ...
##  $ SEX          : Factor w/ 2 levels "F","M": 1 1 1 2 1 1 1 2 2 1 ...
##  $ SOURCE       : Factor w/ 2 levels "in","out": 2 2 2 2 2 2 2 2 2 2 ...
```

The data description lacks unit specification, yet upon observing the boxplot, it's evident that our predictors are measured in diverse units. To ensure compatibility for PCA, k-means, and hierarchical clustering, we standardize our predictors by scaling them.

```
boxplot(df)
```



We'll exclude both the response variable and the gender variable from our analysis. The specific variables in our dataset aren't anticipated to be influenced by an individual's gender.

```
ds <- df
ds <- ds %>% mutate(SEX=NULL)
ds <- ds %>% mutate(SOURCE=NULL)

ds <-scale(ds,center = TRUE, scale = TRUE)
ds <- data.frame(ds)
View(ds)
```

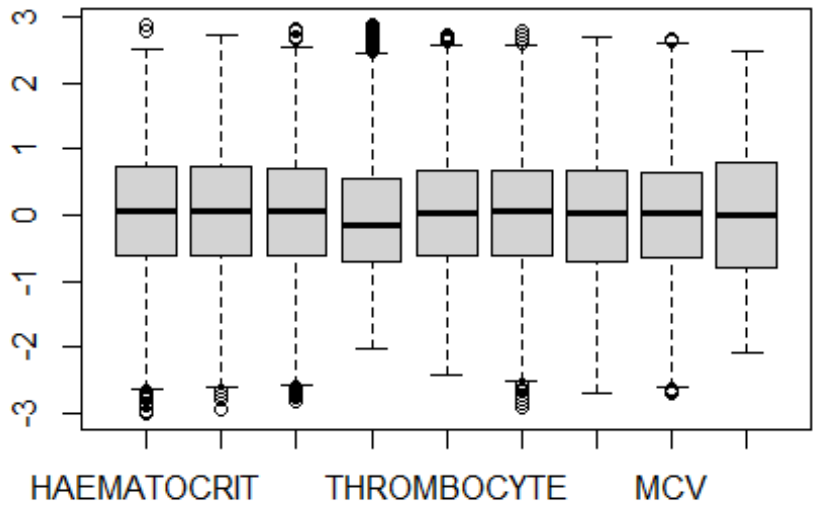
This summary displays the ranges of the scaled variables.

```
summary(ds)
```

```
##   HAEMATOCRIT        HAEMOGLOBINS       ERYTHROCYTE         LEUCOCYTE
##  Min.   :-3.00441   Min.   :-2.93622   Min.   :-2.82812   Min.   :-2.0371
##  1st Qu.:-0.62778   1st Qu.:-0.63012   1st Qu.:-0.61977   1st Qu.:-0.7237
##  Median : 0.05778   Median : 0.05123   Median : 0.05861   Median :-0.1434
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.72506   3rd Qu.: 0.73258   3rd Qu.: 0.69369   3rd Qu.: 0.5591
##  Max.   : 2.86402   Max.   : 2.72422   Max.   : 2.82987   Max.   : 2.8804
##   THROMBOCYTE            MCH                MCHC               MCV
##  Min.   :-2.42780   Min.   :-2.90147   Min.   :-2.7149   Min.   :-2.68791
##  1st Qu.:-0.62635   1st Qu.:-0.63141   1st Qu.:-0.7041   1st Qu.:-0.65590
##  Median : 0.03896   Median : 0.07123   Median : 0.0367   Median : 0.02814
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000
##  3rd Qu.: 0.66332   3rd Qu.: 0.66577   3rd Qu.: 0.6717   3rd Qu.: 0.65182
##  Max.   : 2.72066   Max.   : 2.77368   Max.   : 2.6825   Max.   : 2.66370
##       AGE
##  Min.   :-2.0938329
##  1st Qu.:-0.7914345
##  Median :-0.0006926
##  Mean   : 0.0000000
##  3rd Qu.: 0.7900494
##  Max.   : 2.4645617
```

There is a significant difference between this boxplot and the previous one.

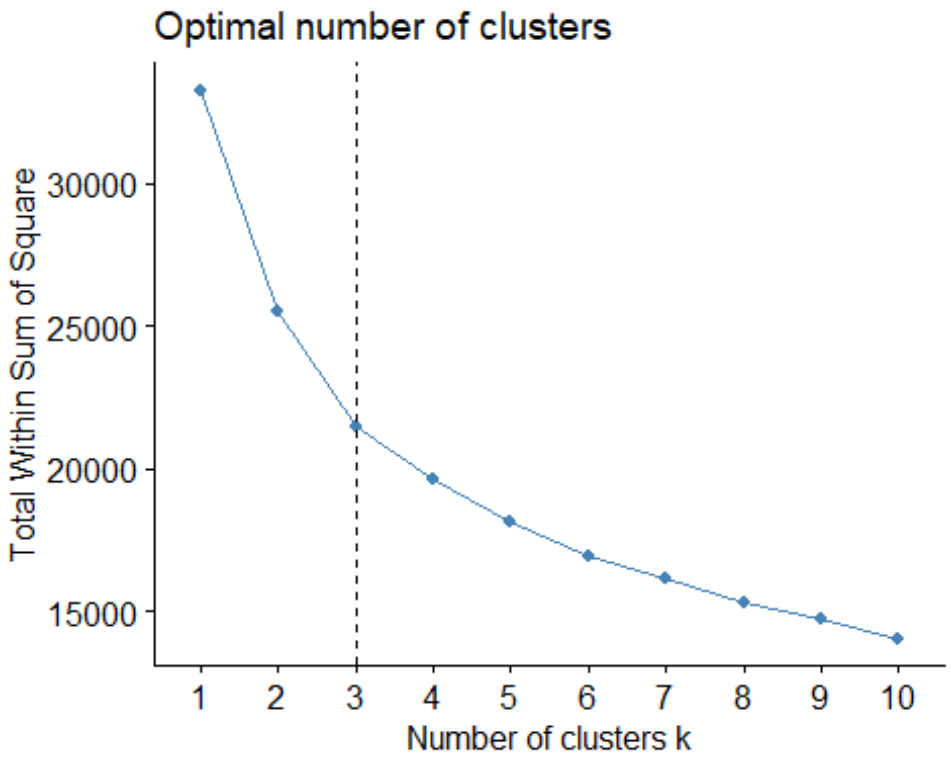
```
boxplot(ds)
```



k-means: We'll employ the elbow technique to select the optimal number of clusters for our K-means analysis We observe that beyond k=3, there's a marginal decrease in the total within Sum of Squares for each k-value. Hence we will generate three clusters

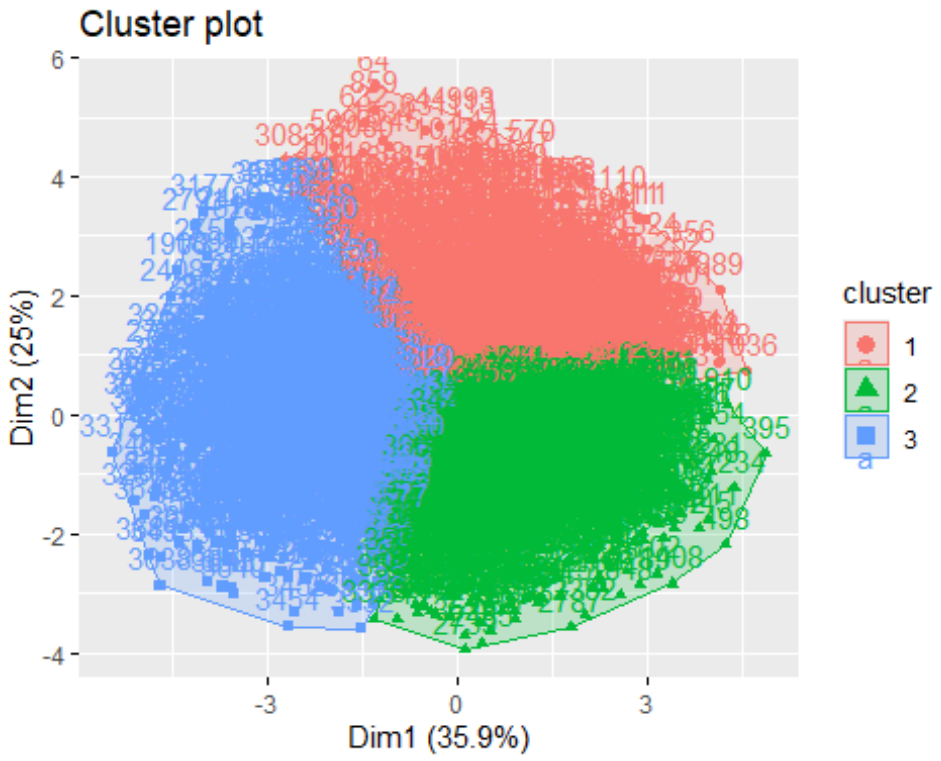
This method operates similarly with k-means as it does with PCA.

```
fviz_nbclust(ds, kmeans, method = "wss") +
  geom_vline(xintercept = 3, linetype = 2)
```



we run the starting random assignment 100 times since the K-means algorithm finds a local rather than a global optimum hence, the results obtained will depend on the initial (random) cluster assignment of each observation

```
fviz_cluster(kmeans(ds,centers=3,iter.max = 10000,nstart=100),data=ds)
```




```
clusters <- kmeans(ds,centers=3,iter.max = 10000,nstart=100)
```

we can see the vector of the p feature means for the observations in the three clusters

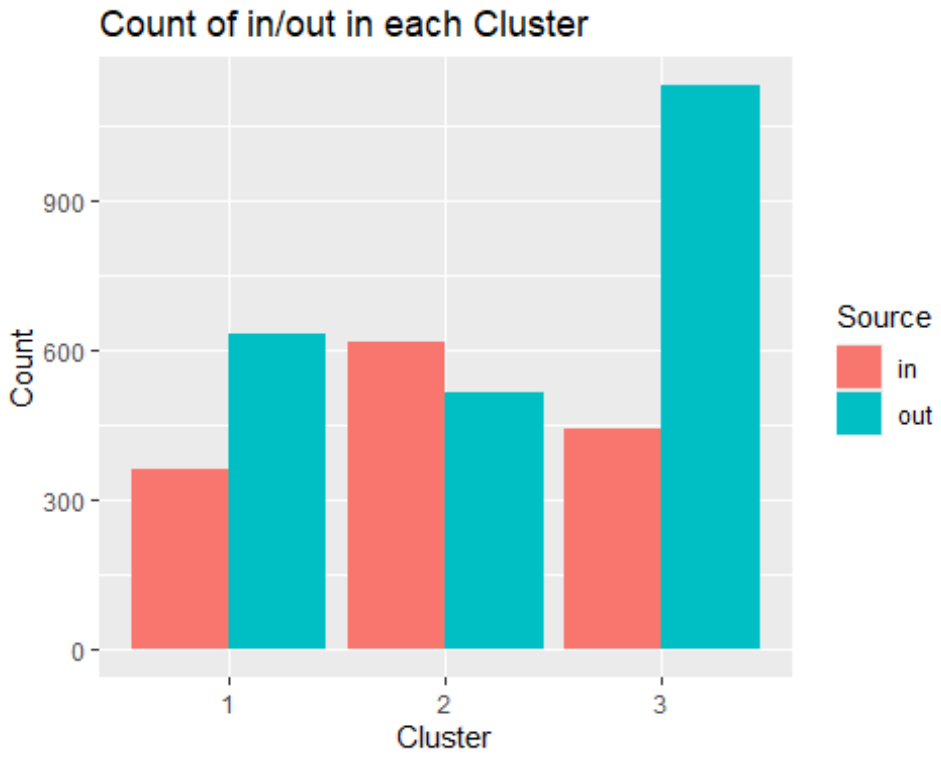
```
clusters$centers
```

```
##   HAEMATOCRIT HAEMOGLOBINS ERYTHROCYTE  LEUCOCYTE THROMBOCYTE        MCH
## 1  -0.0571521   -0.1584120   0.3685993 -0.1342649  0.09174203 -1.1960454
## 2  -1.0068867   -0.9842641  -1.1312703  0.3087241 -0.01578553  0.4868009
## 3   0.7586370    0.8064322   0.5786802 -0.1366295 -0.04668289  0.4069705
##          MCHC        MCV         AGE
## 1 -0.52259354 -1.0883576 -0.76972160
## 2 -0.08451907  0.5883594  0.69541271
## 3  0.39109127  0.2660038 -0.01229152
```

we reinclude the SOURCE variable previously removed In clusters one and three, the count of out-care patients exceeds that of in-care patients, while in cluster two, the reverse is observed.

However, the response here is not very significant for our data analysis

```
ds <- ds %>% mutate(cluster= clusters$cluster)
ds <- ds %>% mutate(SOURCE= df$SOURCE)
ggplot(ds, aes(x = as.factor(cluster), fill = as.factor(SOURCE))) +
  geom_bar(position = "dodge") +
  labs(x = "Cluster", y = "Count", fill = "Source") +
  ggtitle("Count of in/out in each Cluster")
```



Hierarchical clustering:

```
ds <- ds %>% mutate(cluster=NULL) %>% mutate(SOURCE=NULL)
```

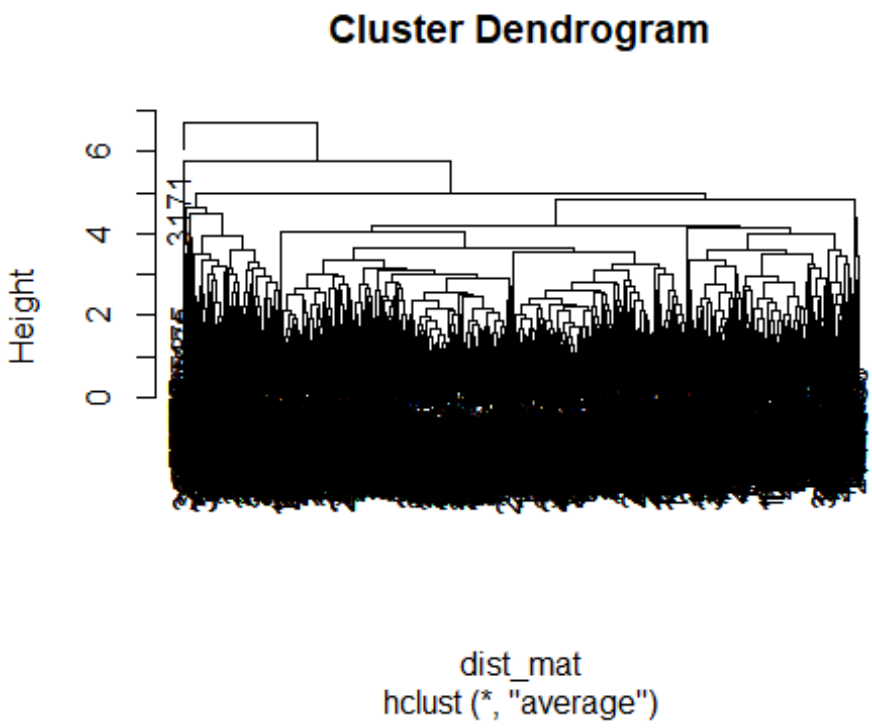
we will use the euclidean distance in our work

```
dist_mat <- dist(ds, method = 'euclidean')
```

We build three trees using average link, single link and complete link respectively.

average linkage

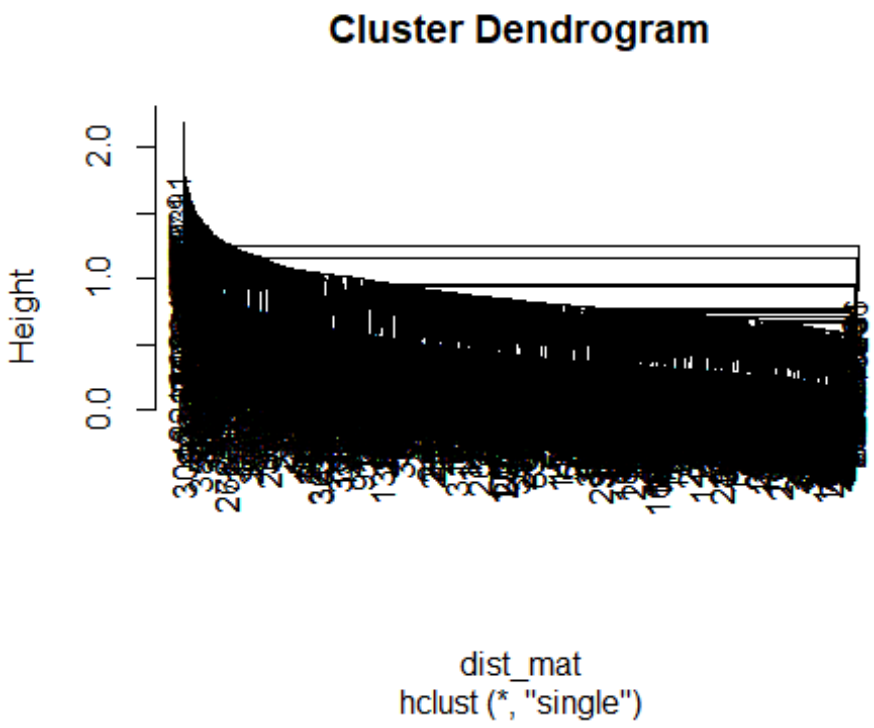
```
hclust_avg <- hclust(dist_mat, method = 'average')
plot(hclust_avg)
```



we can conclude that observation 3171 is very different from any other observation, since it did not fuse with any other leaf or branch, and it is found at the top of the tree

single linkage

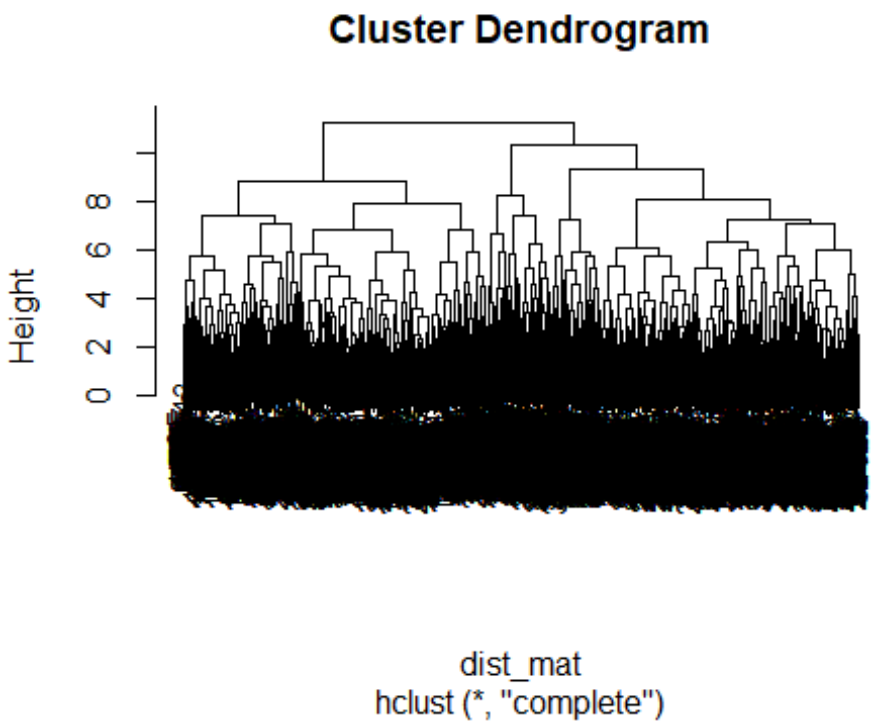
```
hclust_single <- hclust(dist_mat, method = 'single')
plot(hclust_single)
```



The result of single linkage are extended, trailing clusters in which single observations are fused one-at-a-time, which is an issue with this method

complete linkage

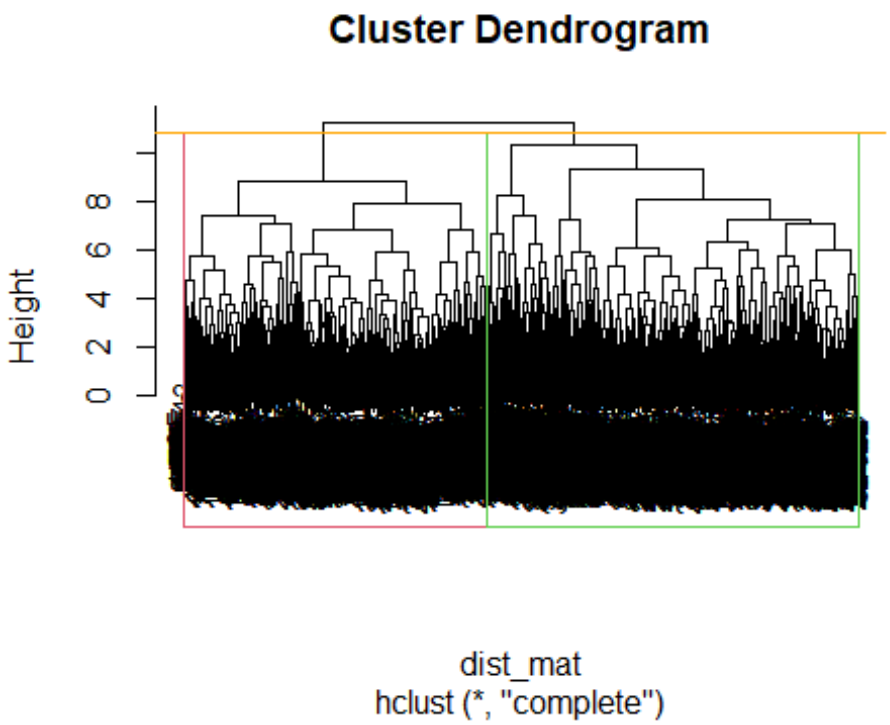
```
hclust_comp <- hclust(dist_mat, method = 'complete')
plot(hclust_comp)
```



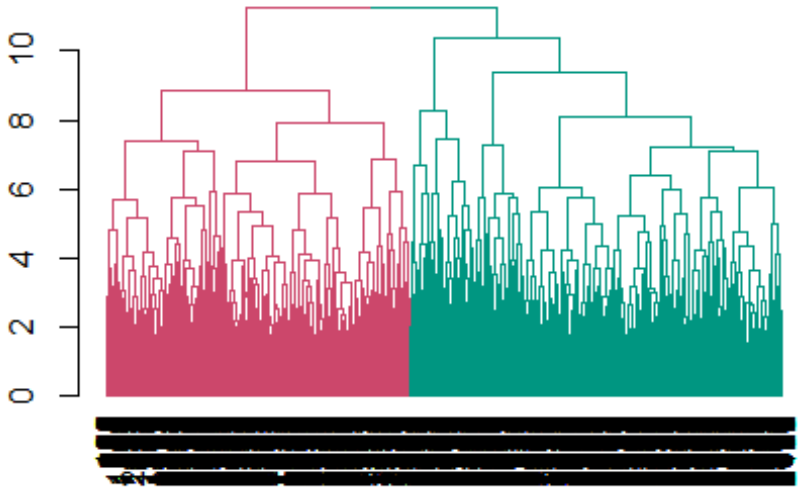
The dendrogram generated using complete linkage appears to be the most balanced, and we'll proceed with this tree for the subsequent analysis.

We can visualize distinct clusters by cutting the tree at various heights. k= number of clusters k=2

```
cut_comp <- cutree(hclust_comp, k = 2)
plot(hclust_comp)
rect.hclust(hclust_comp , k = 2, border = 2:6)
abline(h = 10.8, col = 'orange')
```

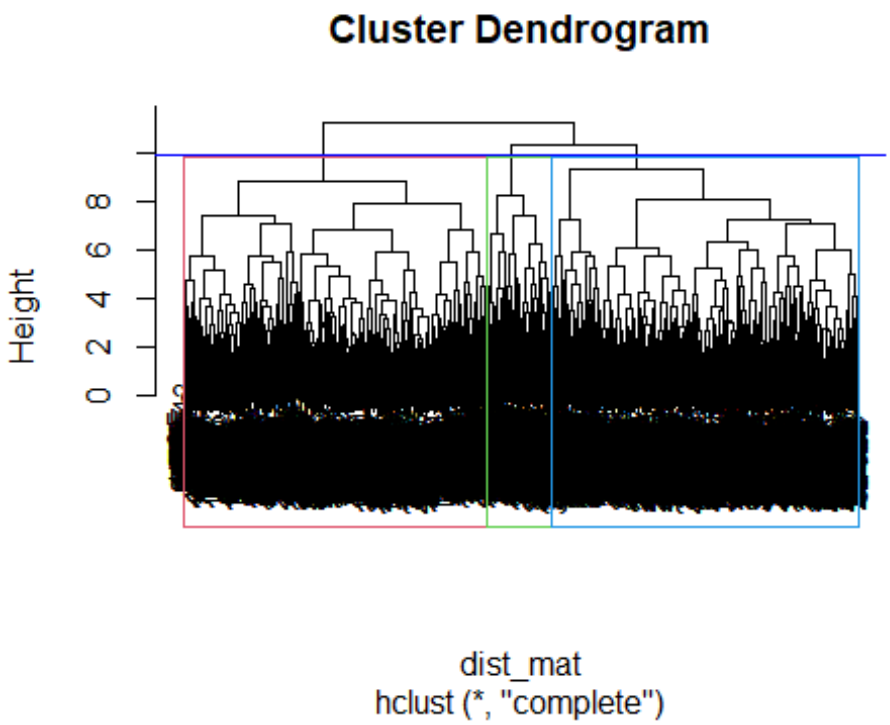



```
comp_dend_obj <- as.dendrogram(hclust_comp)
comp_col_dend <- color_branches(comp_dend_obj, h = 10.8)

## Loading required namespace: colorspace

plot(comp_col_dend)
```

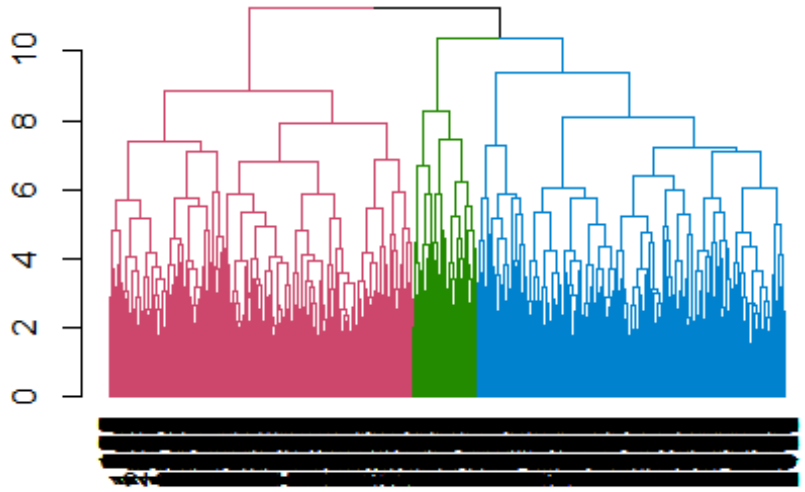


k=3

```
cut_comp1 <- cutree(hclust_comp, k = 3)
plot(hclust_comp)
rect.hclust(hclust_comp , k = 3, border = 2:6)
abline(h = 9.9, col = 'blue')
```

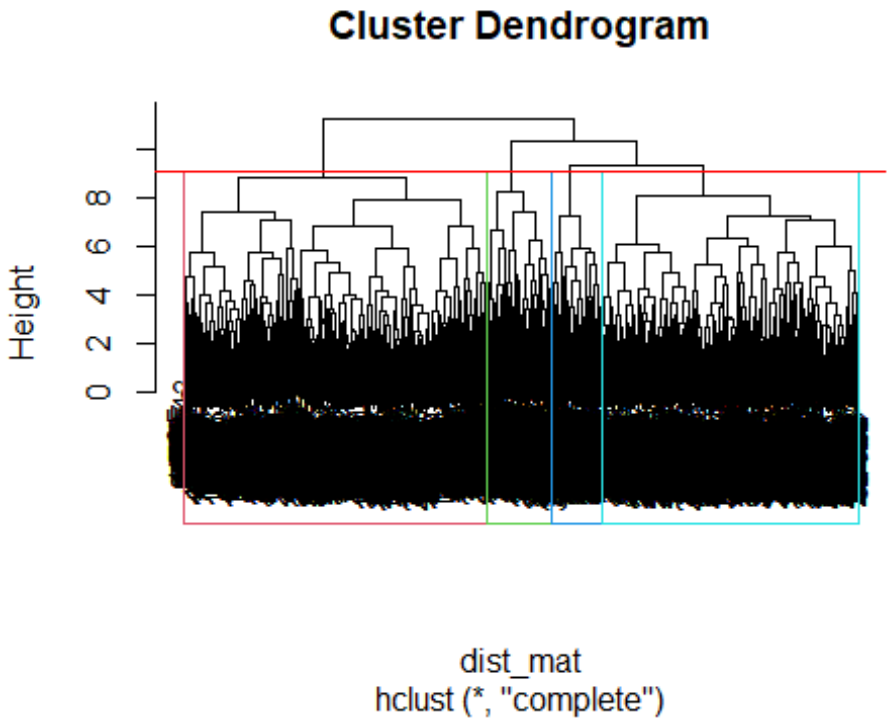



```r
comp_dend_obj <- as.dendrogram(hclust_comp)
comp_col_dend <- color_branches(comp_dend_obj, h = 9.9)
plot(comp_col_dend)
```

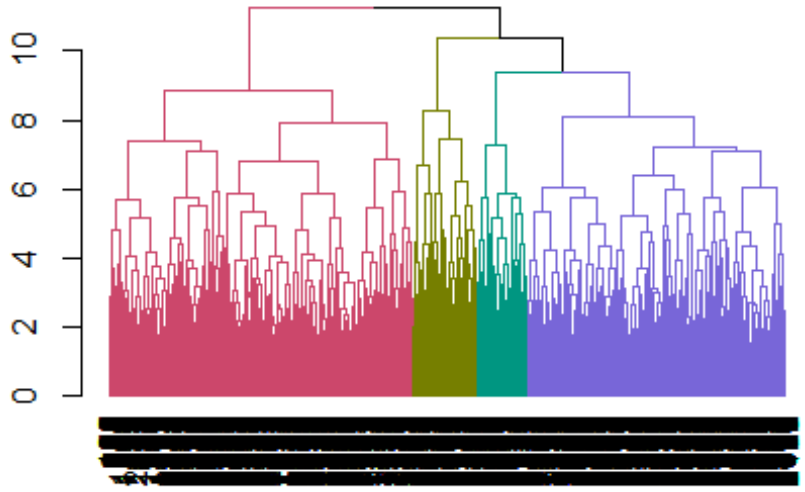


k=4

```
cut_comp2 <- cutree(hclust_comp, k = 4)
plot(hclust_comp)
rect.hclust(hclust_comp , k = 4, border = 2:6)
abline(h = 9.1, col = 'red')
```

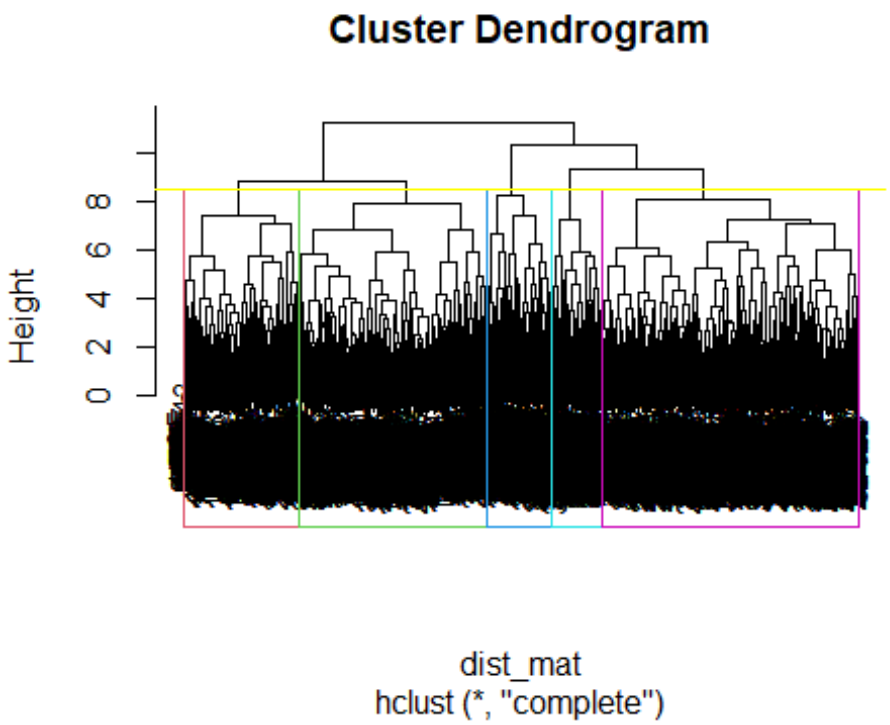



```
comp_dend_obj <- as.dendrogram(hclust_comp)
comp_col_dend <- color_branches(comp_dend_obj, h = 9.1)
plot(comp_col_dend)
```

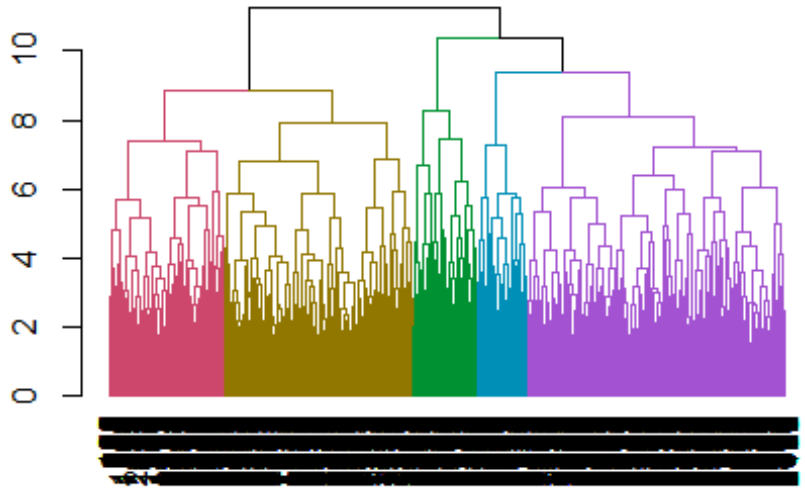


k=5

```
cut_comp3 <- cutree(hclust_comp, k = 5)
plot(hclust_comp)
rect.hclust(hclust_comp , k = 5, border = 2:6)
abline(h = 8.5, col = 'yellow')
```




```
comp_dend_obj <- as.dendrogram(hclust_comp)
comp_col_dend <- color_branches(comp_dend_obj, h = 8.5)
plot(comp_col_dend)
```



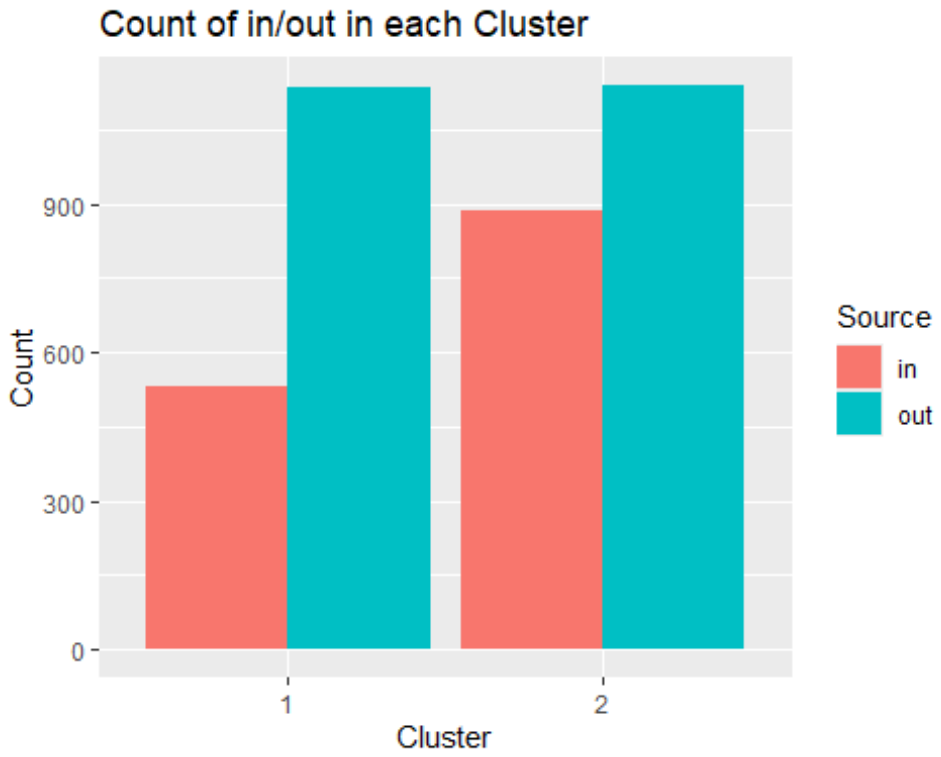
Visually, the most sensible segmentation of the dendrogram appears to be cutting it into two clusters at a height of h=10.8. Lower heights result in smaller clusters that don't seem to hold substantial individual significance, suggesting that integrating them into larger clusters would make more sense.

Hence using K-means clustering results in three clusters, and using hierarchical clustering results in two clusters.

```
ds <- ds %>% mutate(cluster = cut_comp)
ds <- ds %>% mutate(SOURCE = df$SOURCE)
```

Since the hierarchical clustering results in two clusters it would make more sense in this case to examine if the majority of in-care patients predominantly belong to one cluster while the out-care patients largely populate the second cluster.

```
ggplot(ds, aes(x = as.factor(cluster), fill = as.factor(SOURCE))) +
  geom_bar(position = "dodge") +
  labs(x = "Cluster", y = "Count", fill = "Source") +
  ggtitle("Count of in/out in each Cluster")
```



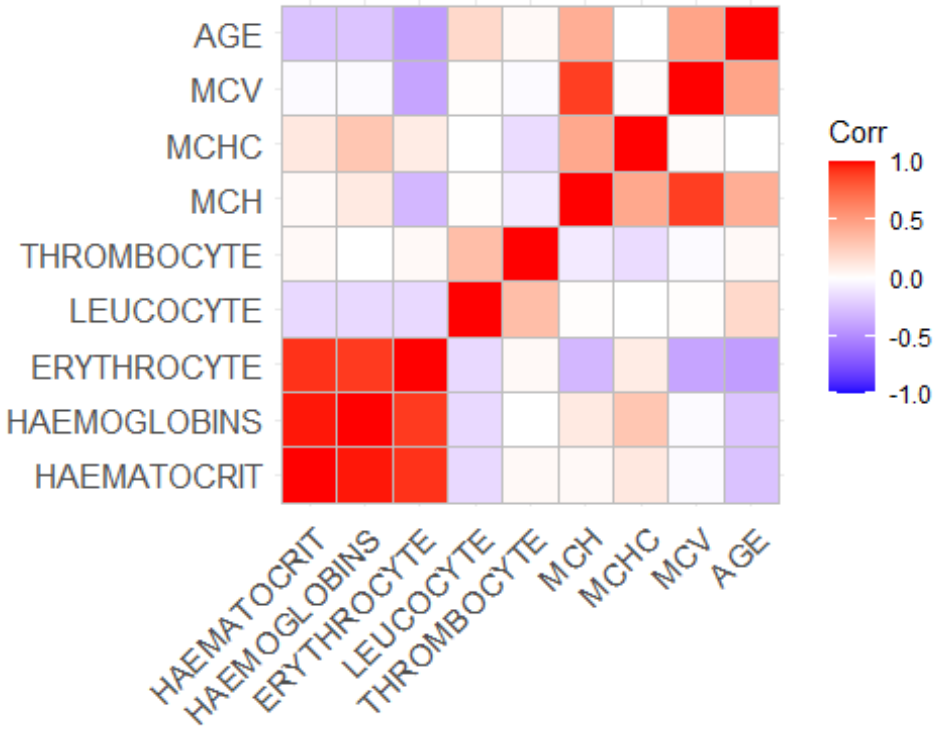
In both clusters, the count of out-care patients surpasses that of in-care patients. It appears that these clusters do not distinctly separate the two types of patients.

PCA

```
ds <- ds %>% mutate(cluster=NULL) %>% mutate(SOURCE=NULL)
```

Given this correlation matrix, we would anticipate “ERYTHROCYTE,” “HAEMOGLOBINS,” and “HAEMATOCRIT” to appear close to each other in the biplot and share a similar directional trend. as well as “MCV” and “MCH”.

```
corr_matrix <- cor(ds)
ggcorrplot(corr_matrix)
```



we notice that nine principal components have been generated (Comp.1 to Comp.9) along with the nine principal component loading vectors, which also correspond to the number of variables in the data. There are at most min(n − 1, p) principal components.

```
data.pca <- princomp(corr_matrix)
data.pca$loadings[, 1:9]
```

```
##                    Comp.1      Comp.2      Comp.3      Comp.4      Comp.5
## HAEMATOCRIT    0.44309654  0.18497362  0.25468212  0.02069490  0.15684678
## HAEMOGLOBINS   0.43037382  0.23701842  0.09635632 -0.01788052  0.11889449
## ERYTHROCYTE    0.52776398  0.02343051  0.08297254  0.11277351  0.04022207
## LEUCOCYTE     -0.16049241 -0.43125068 -0.22274881  0.09565462  0.77845686
## THROMBOCYTE   -0.02685981 -0.47093160  0.23176265 -0.65260668 -0.31963477
## MCH           -0.27780000  0.49120040  0.02770476 -0.29968916  0.17930887
## MCHC           0.02597987  0.31495405 -0.78233081 -0.19803127 -0.16958444
## MCV           -0.32298089  0.39646750  0.41068691 -0.24011093  0.28355389
## AGE           -0.36230555  0.06036517  0.18483583  0.60399142 -0.33191946
##                    Comp.6       Comp.7        Comp.8       Comp.9
## HAEMATOCRIT   0.24054646  0.327370947  7.148436e-01  0.059349010
## HAEMOGLOBINS  0.29529475  0.383608008 -6.375902e-01 -0.306569851
## ERYTHROCYTE   0.19388798 -0.763297539 -1.120115e-01  0.258715699
## LEUCOCYTE     0.35140934 -0.003488453  7.733159e-05  0.001553854
## THROMBOCYTE   0.44233313 -0.007620893 -6.448505e-04  0.003017492
## MCH           0.17655962 -0.388836879  1.721078e-01 -0.588921029
## MCHC          0.33153132  0.089231398  4.536354e-02  0.316510851
## MCV           0.03447816  0.061692607 -1.955783e-01  0.623322901
## AGE           0.59659340 -0.004176122 -6.433896e-04  0.002958501
```

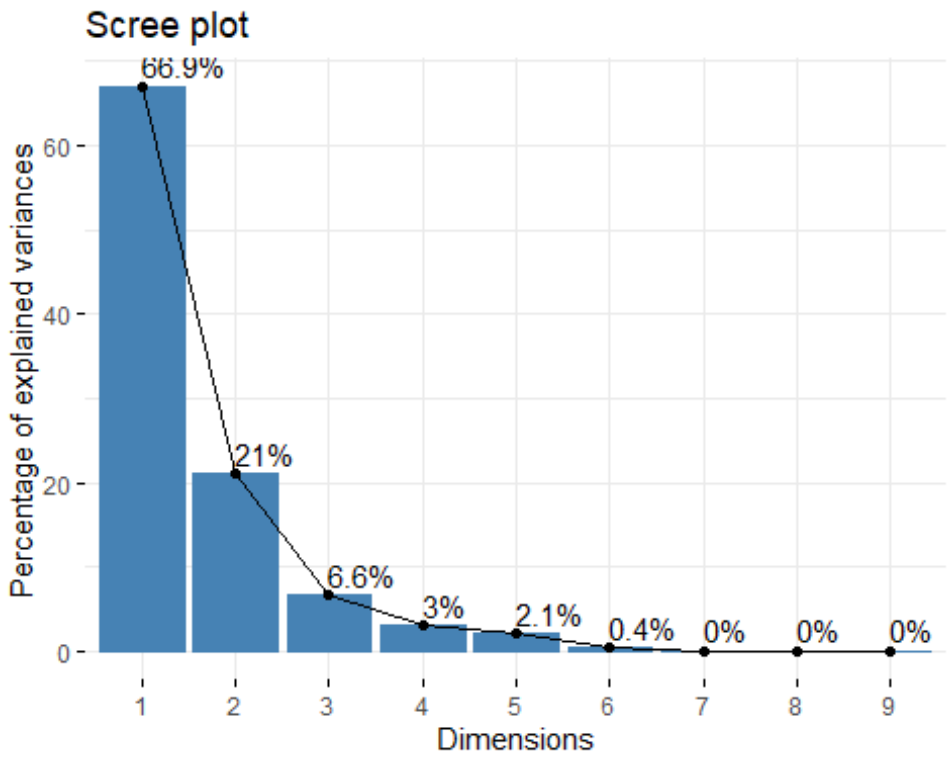
Each component explains a percentage of the total variance in the data set. In the Cumulative Proportion section, the first principal component explains almost 67% of the total variance. The second one explains 21% of the total variance. The cumulative proportion of Comp.1 and Comp.2 explains nearly 88% of the total variance. This means that the first two principal components can accurately represent the data.

```
summary(data.pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4
Comp.5
## Standard deviation     1.0590958 0.5932478 0.33367115 0.22491669
0.18581177
## Proportion of Variance 0.6691248 0.2099466 0.06641618 0.03017727
0.02059601
## Cumulative Proportion  0.6691248 0.8790714 0.94548754 0.97566481
0.99626081
##                             Comp.6       Comp.7       Comp.8 Comp.9
## Standard deviation     0.079170271 4.684142e-04 1.301293e-04      0
## Proportion of Variance 0.003739046 1.308870e-07 1.010153e-08      0
## Cumulative Proportion  0.999999859 1.000000e+00 1.000000e+00      1
```

After PC2 the proportion of variance explained by each subsequent principal component drops off. According to the elbow technique we will choose the first two components

```
fviz_eig(data.pca, addlabels = TRUE)
```

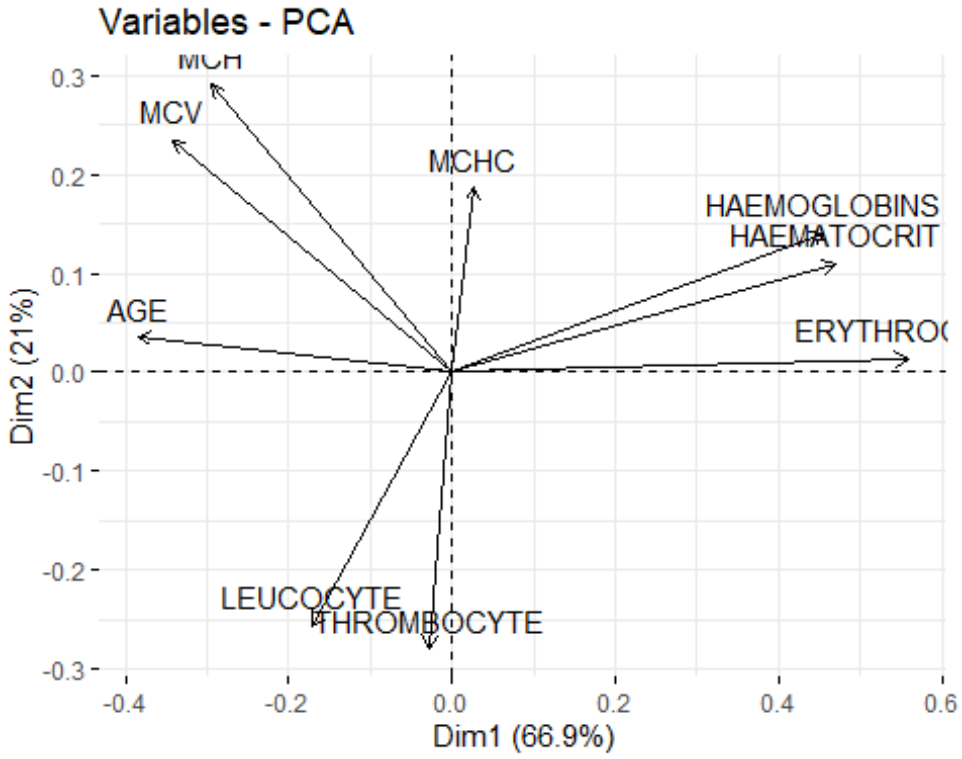


The first principal component places most of its wait on all predictors except “MCHC”, “LEUCOCYE” and “THROMBOCYE”. The second principal component places most of its wait on all predictors except “HAEMATOCRIT”,“HAEMOGLOBINS”, “ERYTHROCYTE”, and “AGE”.

```
data.pca$loadings[, 1:2]
```

```
##                    Comp.1      Comp.2
## HAEMATOCRIT   0.44309654  0.18497362
## HAEMOGLOBINS  0.43037382  0.23701842
## ERYTHROCYTE   0.52776398  0.02343051
## LEUCOCYTE    -0.16049241 -0.43125068
## THROMBOCYTE  -0.02685981 -0.47093160
## MCH          -0.27780000  0.49120040
## MCHC          0.02597987  0.31495405
## MCV          -0.32298089  0.39646750
## AGE          -0.36230555  0.06036517
```

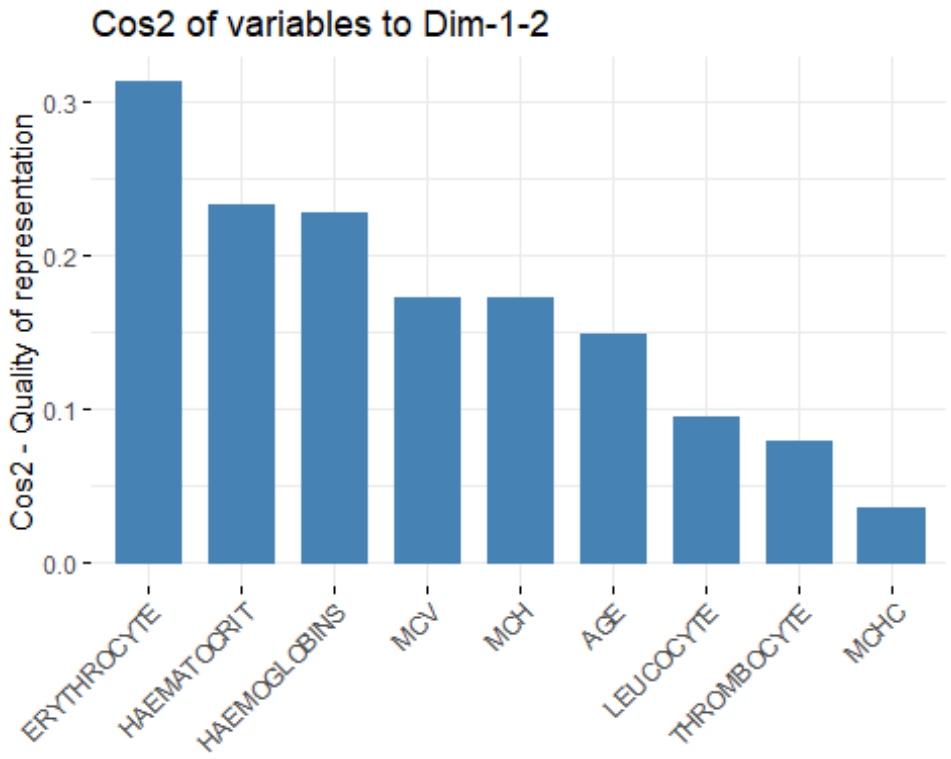
```
fviz_pca_var(data.pca, col.var = "black")
```



The conclusions made after visualizing the correlation matrix are now seen in this biplot.

This plot determine how much each variable is represented in the first two components.

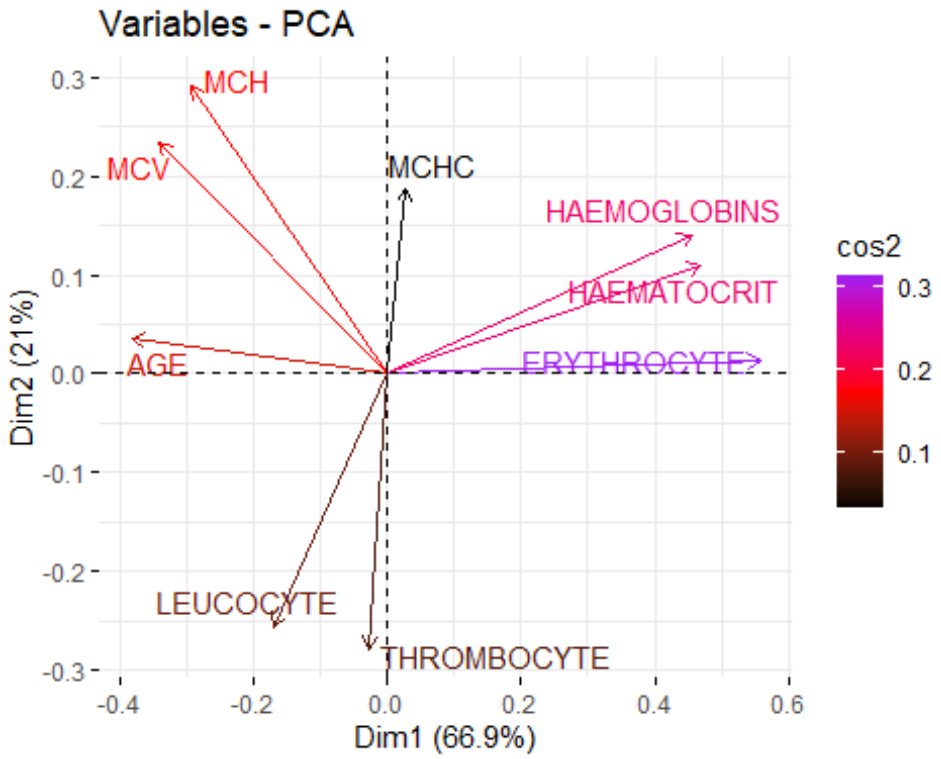
```
fviz_cos2(data.pca, choice = "var", axes = 1:2)
```



"ERYTHROCYTE" is contributing the most to PC1 and PC2, followed by "HAEMATOCRIT" and "HAEMOGLOBINS". "LEUCOCYTE"," THROMBOCYTE" and "MCHC" are not perfectly represented by these components.

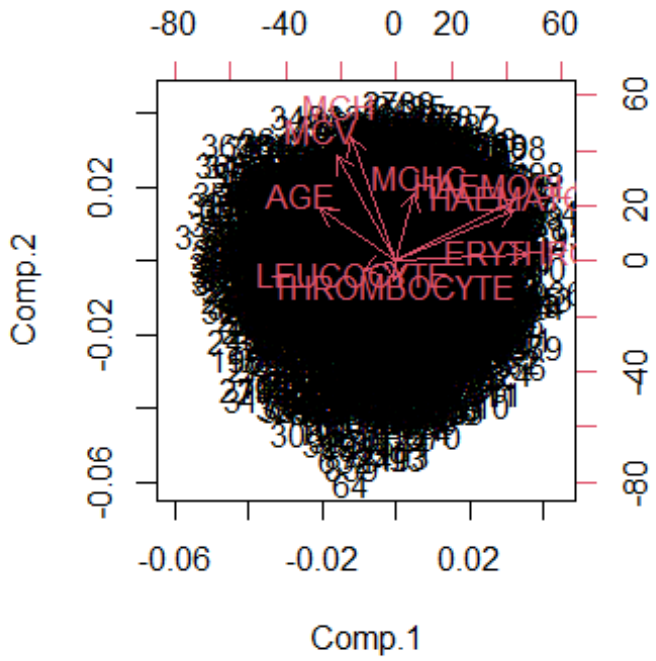
The last two visualization approaches: biplot and attributes importance can be combined to create a single biplot

```
fviz_pca_var(data.pca, col.var = "cos2",
             gradient.cols = c("black", "red", "purple"),
             repel = TRUE)
```



We are not interested in plotting the observations, since the plot will not generate any useful information about the observations

```
results <- princomp(ds)
biplot(results)
```



Classification tree First we split our data into training and testing to perform all the necessary decision trees

```
set.seed(123)
sample= sample.split(df$SOURCE, SplitRatio = .70)
train= subset(df,sample==TRUE)
test= subset(df,sample==FALSE)
```

We train the tree model using the training data and make predictions using the testing dataset.

```
tree <- rpart(SOURCE ~., data = train)
patient.type.predicted <- predict(tree, test, type="class")

confusionMatrix(patient.type.predicted, test$SOURCE)
```

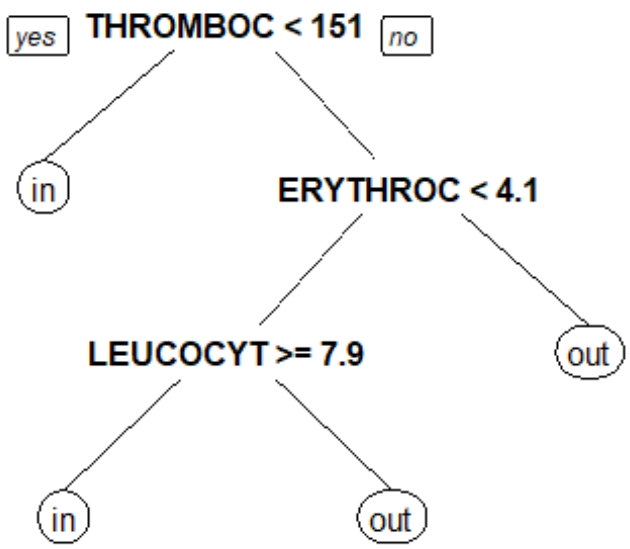
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  in out
##        in  240  99
##        out 185 584
##
##                Accuracy : 0.7437
##                  95% CI : (0.7169, 0.7692)
##     No Information Rate : 0.6164
##     P-Value [Acc > NIR] : < 2.2e-16
```

```
##
##                   Kappa : 0.4364
##
##  Mcnemar's Test P-Value : 4.563e-07
##
##             Sensitivity : 0.5647
##             Specificity : 0.8551
##          Pos Pred Value : 0.7080
##          Neg Pred Value : 0.7594
##              Prevalence : 0.3836
##          Detection Rate : 0.2166
##    Detection Prevalence : 0.3060
##       Balanced Accuracy : 0.7099
##
##        'Positive' Class : in
##
```

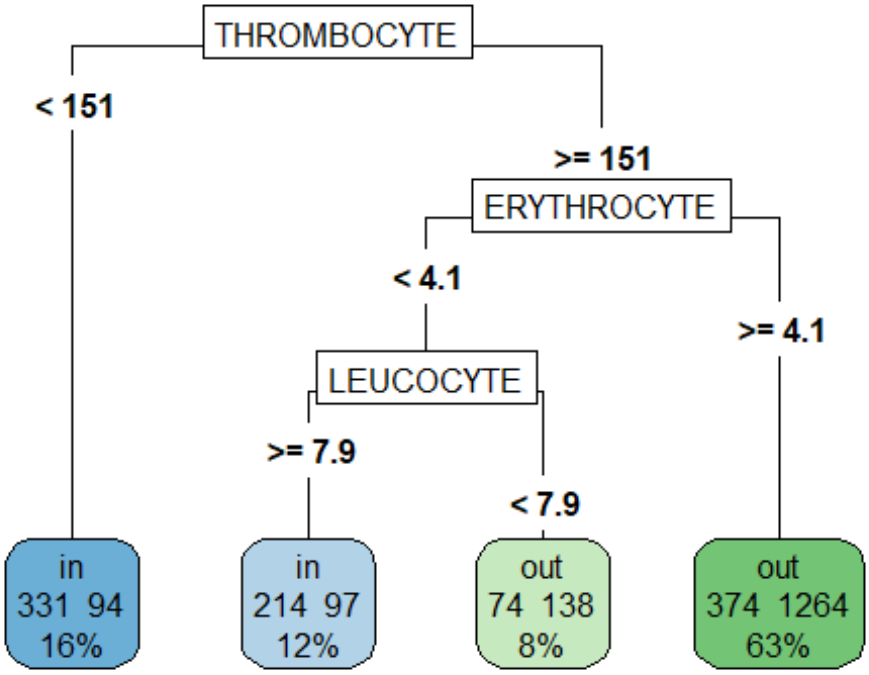
Our classification tree has an accuracy of 0.7437, sensitivity of 0.5647 and specificity of 0.8551 A reasonably good result for an unpruned tree, and without employing any methods to enhance prediction accuracy. One can expect worst results due to overfitting.

We present two visualization for our tree, the first is a simple one

```
prp(tree)
```




```
rpart.plot(tree, type = 5, extra = 101, under = FALSE, cex = 1, box.palette =
"auto")
```

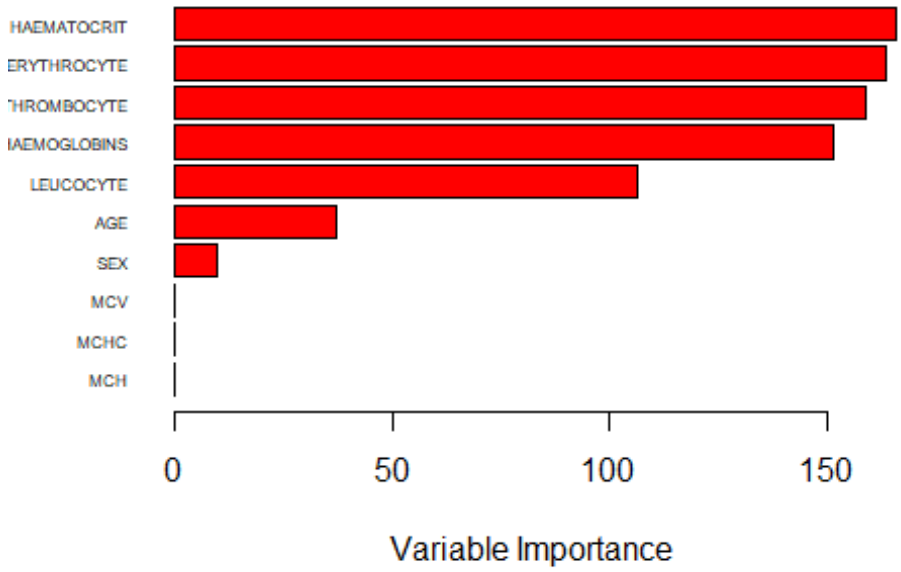



```
rules <- rpart.rules(tree)
print(rules)
```

```
##  SOURCE
##    0.22 when THROMBOCYTE <  151
##    0.31 when THROMBOCYTE >= 151 & ERYTHROCYTE <  4.1 & LEUCOCYTE >= 7.9
##    0.65 when THROMBOCYTE >= 151 & ERYTHROCYTE <  4.1 & LEUCOCYTE <  7.9
##    0.77 when THROMBOCYTE >= 151 & ERYTHROCYTE >= 4.1
```

Surprisingly, the model's complexity is lower than anticipated. Out of the ten predictors, the model utilized only three variables.

```
VI_tree<- data.frame(var = names(df)[-1], imp = varImp(tree))
VI_plot_tree <- VI_tree[order(VI_tree$Overall, decreasing = FALSE), ]
barplot(VI_plot_tree$Overall,
        names.arg = rownames(VI_plot_tree),
        horiz = TRUE,
        col = 'red',
        cex.names = 0.5,
        las = 1,
        xlab = 'Variable Importance')
```




```
varImp(tree)
```

```
##                  Overall
## AGE            37.348065
## ERYTHROCYTE   163.200816
## HAEMATOCRIT   165.752882
## HAEMOGLOBINS  151.435400
## LEUCOCYTE     106.089656
## SEX             9.948225
## THROMBOCYTE   158.569315
## MCH             0.000000
## MCHC            0.000000
## MCV             0.000000
```

It's notable that MCH, MCV, and MCHC held no significance in constructing our tree. Interestingly, the tree didn't utilize the most important feature, "HAEMATOCRIT". It used the second "ERYTHROCYTE" , the third "THROMBOCYTE" and the fifth "LEUCOCYTE" most important variables From the correlation matrix previously generated, a correlation between "ERYTHROCYTE," "HAEMATOCRIT," and "HAEMOGLOBINS" was shown. We suspect that the tree selected "ERYTHROCYTE", due to its lower Gini index among these correlated predictors, then proceeded to use the other two most important variables that weren't part of this correlated set.

Pruning the tree

Given the tree's current simplicity, we anticipate that the pruned tree may either remain unchanged or undergo only marginal changes.

We use Cp – Complexity parameter to control the tree growth. Any split which does not improve the fit by cp will likely be pruned off to avoid overfitting. We want to choose cp that give us the lowest test error

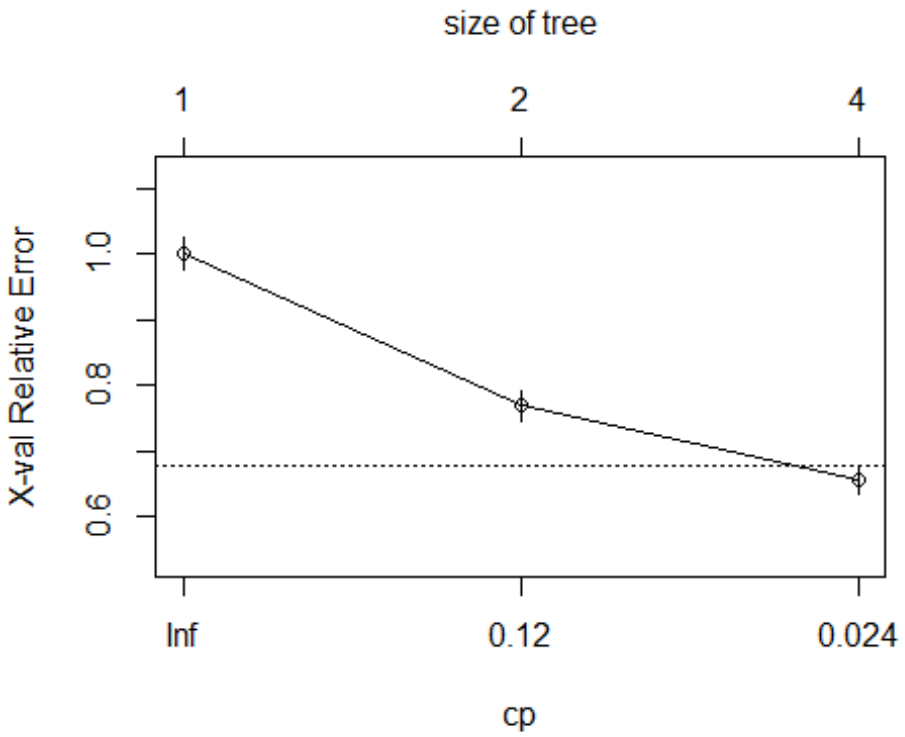
This following function shows the Training error , cross validation error (xerror: the one of interest) and standard deviation at each node of our tree.

```
printcp(tree)
```

```
##
## Classification tree:
## rpart(formula = SOURCE ~ ., data = train)
##
## Variables actually used in tree construction:
## [1] ERYTHROCYTE LEUCOCYTE   THROMBOCYTE
##
## Root node error: 993/2586 = 0.38399
##
## n= 2586
##
##         CP nsplit rel error  xerror     xstd
## 1 0.238671      0   1.00000 1.00000 0.024907
## 2 0.058912      1   0.76133 0.76939 0.023365
## 3 0.010000      3   0.64350 0.65458 0.022215
```

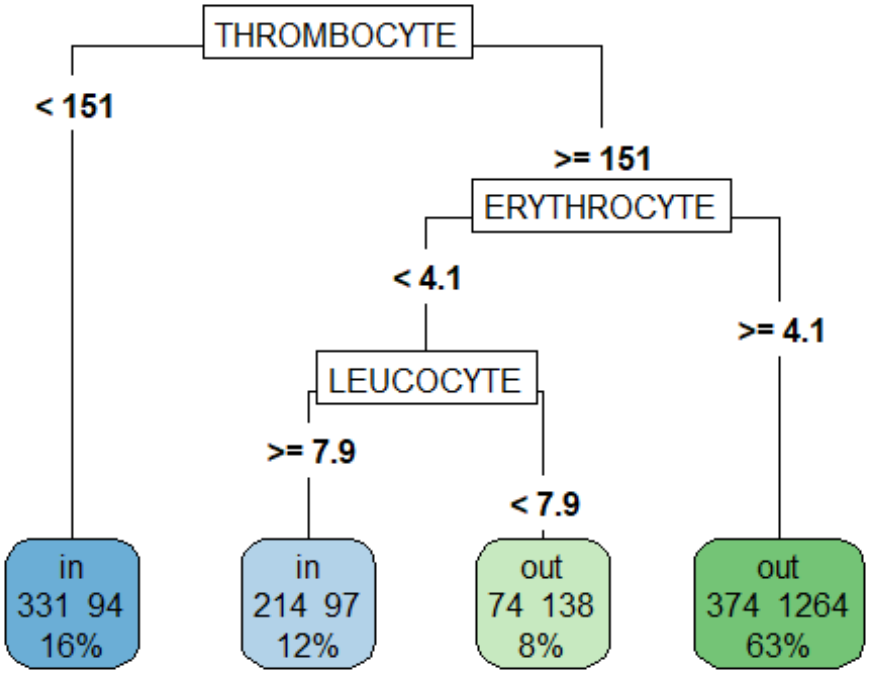
cross-validation results

```
plotcp(tree)
```



We get cp with minimun cross validation error

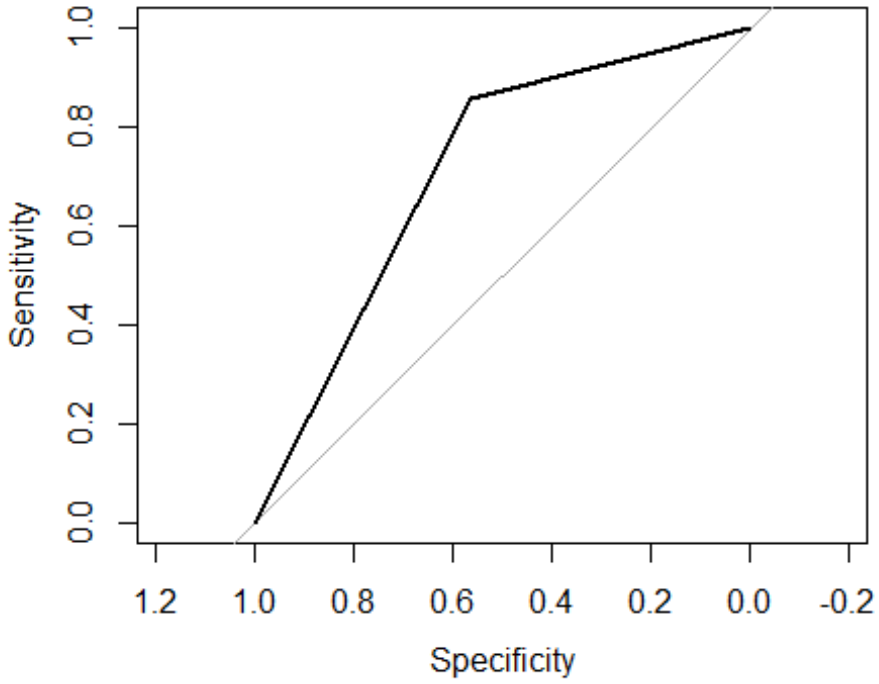
```
best_cp <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
best_cp

## [1] 0.01

tree.pruned <- prune(tree, cp = best_cp)
rpart.plot(tree.pruned, type = 5, extra = 101, under = FALSE, cex = 1,
box.palette = "auto")
```



As expected the tree do not change, we do not expect overfitting in such a simple tree. Removing too many nodes will reduce acuuracy.

ROC and area under the curve “0.7099”.

```
rocdf <- df
rocdf$SOURCE <- ifelse(rocdf$SOURCE=="in",0,1)
patient.type.predicted <- ifelse(patient.type.predicted=="in",0,1)
roc= roc(response=test$SOURCE, predictor= patient.type.predicted)

## Setting levels: control = in, case = out

## Setting direction: controls < cases

auc(roc)

## Area under the curve: 0.7099

plot(roc)
```



For now our classification tree has an accuracy of 0.7437. Using bagging, random forest and boosting, we see if we can improve our prediction accuracy.

Bagging

```
set.seed(123)
patient_bag <- randomForest(formula=SOURCE~., data=train, mtry=(ncol(train)-
1),
                            importance=T, ntree=100)
patient_bag

##
## Call:
##  randomForest(formula = SOURCE ~ ., data = train, mtry = (ncol(train) -
1), importance = T, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 10
##
##         OOB estimate of  error rate: 24.21%
## Confusion matrix:
##      in  out class.error
## in  602  391   0.3937563
## out 235 1358   0.1475204
```

The OOB estimate of error rate is = 0.2421.

```
patient_bag_pred <- predict(object=patient_bag, newdata=test, type="class")
table(test$SOURCE, patient_bag_pred)
```

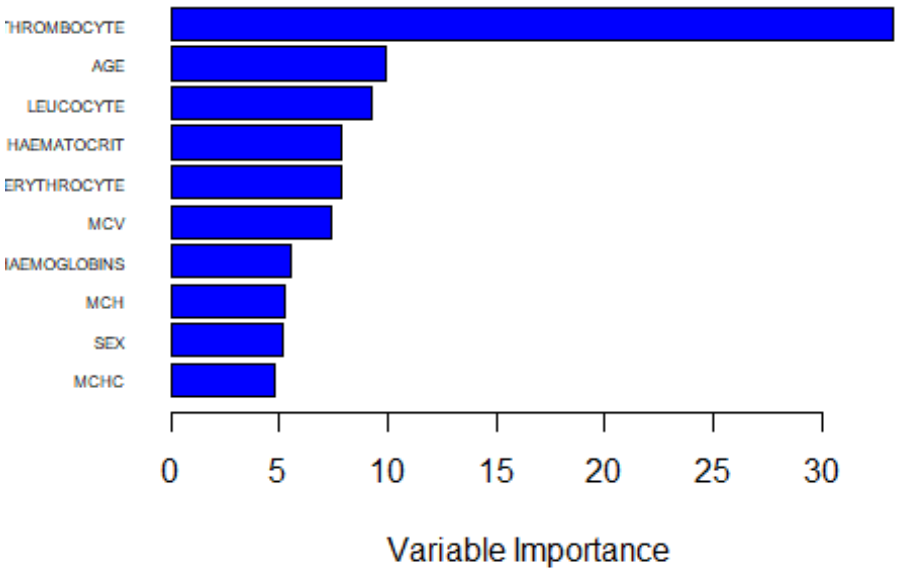
```
##      patient_bag_pred
##        in out
##   in  263 162
##   out  97 586
```

```
acc_bag <- mean(test$SOURCE==patient_bag_pred)
acc_bag
```

```
## [1] 0.7662455
```

The bagging approach generated better accuracy “0.7644404” from the previous tree model “0.7437”.

```
VI<- data.frame(var = names(df)[-1], imp = varImp(patient_bag))
VI_plot_bag <- VI[order(VI$imp.in, decreasing = FALSE), ]
barplot(VI_plot_bag$imp.in,
        names.arg = rownames(VI_plot_bag),
        horiz = TRUE,
        col = 'blue',
        cex.names = 0.5,
        las = 1,
        xlab = 'Variable Importance')
```




```
varImp(patient_bag)
```

```
##                      in       out
## HAEMATOCRIT   7.834535  7.834535
## HAEMOGLOBINS  5.484114  5.484114
## ERYTHROCYTE   7.828268  7.828268
## LEUCOCYTE     9.264727  9.264727
## THROMBOCYTE  33.303979 33.303979
## MCH           5.233584  5.233584
## MCHC          4.815317  4.815317
## MCV           7.399079  7.399079
## AGE           9.875950  9.875950
## SEX           5.169605  5.169605
```

The plot depicting variable importance is presenting results that differ from those shown in the previous plot.

Random forest

```
set.seed(123)
patient_rf <- randomForest(formula=SOURCE~., data=train,
mtry=sqrt(ncol(train)-1),
                           importance=T, ntree=100)
patient_rf

##
## Call:
##  randomForest(formula = SOURCE ~ ., data = train, mtry = sqrt(ncol(train)
-      1), importance = T, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 24.48%
## Confusion matrix:
##      in  out class.error
## in  599  394   0.3967774
## out 239 1354   0.1500314
```

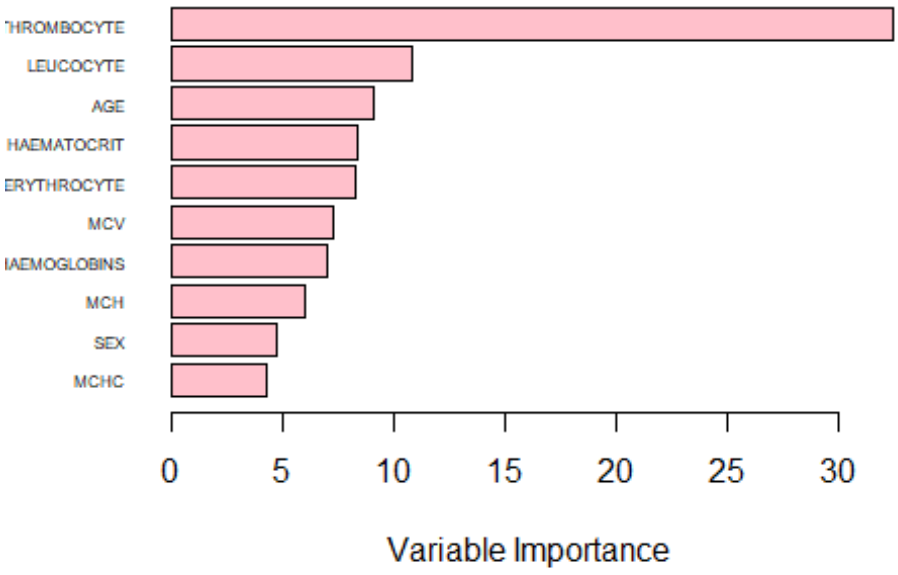
The OOB estimate of error rate is =0.2448 very close but bigger than the OOB generated by bagging = 0.2421.

```
patient_rf_pred <- predict(object=patient_rf, newdata=test, type="class")
table(test$SOURCE, patient_rf_pred)

##      patient_rf_pred
##        in out
##   in  265 160
##   out 108 575

acc_rf <- mean(test$SOURCE==patient_rf_pred)
acc_rf
```

```
## [1] 0.7581227
```

The accuracy of the random forest model is “0.7581227” better than the classification tree model “0.7437”, but worse than the bagging approach accuracy “0.7644404”.

```
VI_rf <- data.frame(var = names(df)[-1], imp = varImp(patient_rf))
VI_plot_rf <- VI_rf[order(VI_rf$imp.in, decreasing = FALSE), ]
barplot(VI_plot_rf$imp.in,
        names.arg = rownames(VI_plot_rf),
        horiz = TRUE,
        col = 'pink',
        cex.names = 0.5,
        las = 1,
        xlab = 'Variable Importance')
```




```
varImp(patient_rf)
```

```
##                      in       out
## HAEMATOCRIT   8.394935  8.394935
## HAEMOGLOBINS  7.028961  7.028961
## ERYTHROCYTE   8.314683  8.314683
## LEUCOCYTE    10.782743 10.782743
## THROMBOCYTE  32.450294 32.450294
## MCH           5.986130  5.986130
## MCHC          4.297299  4.297299
## MCV           7.323518  7.323518
```

```
## AGE           9.107194  9.107194
## SEX           4.707256  4.707256
```

Boosting We'll use the caret workflow, which invokes the xgboost package, to automatically adjust the model parameter values, and fit the final best boosted tree that explains the best our data. We do this since we can not choose in boosting a random big value for the number of trees, because it will lead to overfitting, unlike in bagging and boosting. This method takes approximately two minutes to run.

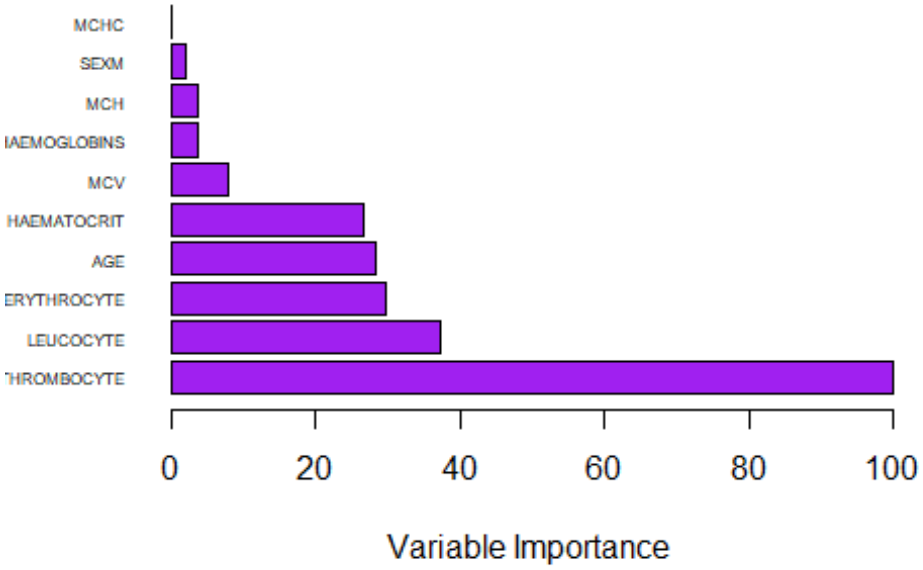
```
set.seed(123)
model <- train(
  SOURCE ~., data = train, method = "xgbTree",
  trControl = trainControl("cv", number = 10)
)

model$bestTune

##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 43      50         3 0.3     0              0.6                1         1

predicted.classes <- model %>% predict(test)
```

The number of boosting trees generated is 50, way less than the number of trees we used in the bagging and boosting methods "1000" , with a max depth of 3.

```
mean(predicted.classes == test$SOURCE)

## [1] 0.7743682
```

The boosting technique produced a model with the highest accuracy among all other approaches, achieving a score of "0.7743682".

```
VI_boost <- as.data.frame(varImp(model)$importance)
VI_plot_boost <- VI_boost[order(VI_boost$Overall, decreasing = TRUE), ]
barplot(VI_plot_boost,
        names.arg = rownames(VI_boost),
        horiz = TRUE,
        col = 'purple',
        cex.names = 0.5,
        las = 1,
        xlab = 'Variable Importance')
```




```
varImp(model)

## xgbTree variable importance
##
##              Overall
## THROMBOCYTE  100.000
## LEUCOCYTE     37.410
## ERYTHROCYTE   29.852
## AGE           28.458
## HAEMATOCRIT   26.666
## MCV            7.938
## HAEMOGLOBINS   3.707
## MCH            3.605
## SEXM           2.089
## MCHC           0.000
```

Variable importance of all used methods in one table

```
my_table <- data.frame(
  "Classification Tree" = c("HAEMATOCRIT", "ERYTHROCYTE", "THROMBOCYTE",
"HAEMOGLOBINS", "LEUCOCYTE", "AGE", "SEX", "MCV", "MCHC", "MCH"),
  "Bagging" = c("THROMBOCYTE", "AGE", "LEUCOCYTE", "HAEMATOCRIT",
"ERYTHROCYTE", "MCV", "HAEMOGLOBINS", "MCH", "SEX", "MCHC"),
  "Random Forest" = c("THROMBOCYTE", "LEUCOCYTE","AGE", "HAEMATOCRIT",
"ERYTHROCYTE", "MCV", "HAEMOGLOBINS", "MCH", "SEX", "MCHC"),
  "Boosting" = c("THROMBOCYTE", "LEUCOCYTE", "ERYTHROCYTE", "AGE",
"HAEMATOCRIT", "MCV", "HAEMOGLOBINS", "MCH", "SEX", "MCHC"),
```

```
  row.names = c("1st", "2nd", "3rd", "4th", "5th", "6th", "7th", "8th",
"9th", "10th")
)
print(my_table)
```

```
##      Classification.Tree      Bagging Random.Forest     Boosting
## 1st          HAEMATOCRIT  THROMBOCYTE   THROMBOCYTE  THROMBOCYTE
## 2nd          ERYTHROCYTE          AGE     LEUCOCYTE    LEUCOCYTE
## 3rd          THROMBOCYTE    LEUCOCYTE           AGE  ERYTHROCYTE
## 4th         HAEMOGLOBINS  HAEMATOCRIT   HAEMATOCRIT          AGE
## 5th            LEUCOCYTE  ERYTHROCYTE   ERYTHROCYTE  HAEMATOCRIT
## 6th                  AGE          MCV           MCV          MCV
## 7th                  SEX HAEMOGLOBINS  HAEMOGLOBINS HAEMOGLOBINS
## 8th                  MCV          MCH           MCH          MCH
## 9th                 MCHC          SEX           SEX          SEX
## 10th                 MCH         MCHC          MCHC         MCHC
```

The categorization of variable importance in bagging, random forest, and boosting is more similar to each other when compared to classification trees. The only difference between bagging and random forest is the swapping of variable importance between “AGE” and “LEUCOCYTES.” The only disparity between random forest and boosting is centered around the variable “ERYTHROCYTE.” In random forest, it ranks fifth in importance, whereas in boosting, it holds the third position. In the classification tree model, no specific level of variable importance categorization matches the equivalent level seen in other methods.