Team Project

José Ignacio Díez Ruiz 100487766
Carlos Roldán Piñero 100484904
Pablo Vidal Fernández 100483812

2023-01-18

Descriptive Analysis and Preprocessing

Reading the data and setting the NAs

Before starting the analysis, we need to make some preprocessing on our data. Let us start by loading it into memory and listing the names of the columns.

```
data <- read.csv("diabetes.csv")
colnames(data)
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```

Of these, our target variable is `Outcome`, which has two levels. For convenience, we transform it into a factor variable which R can trat accordingly.

```
data$Outcome <- factor(data$Outcome, c(0, 1), c("Negative", "Positive"))
```

Now we need to address a particularity of the chosen data: not-a-number (NaN) instances are encoded as zeros in variables where that value is imposible [1]. These are:

- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI

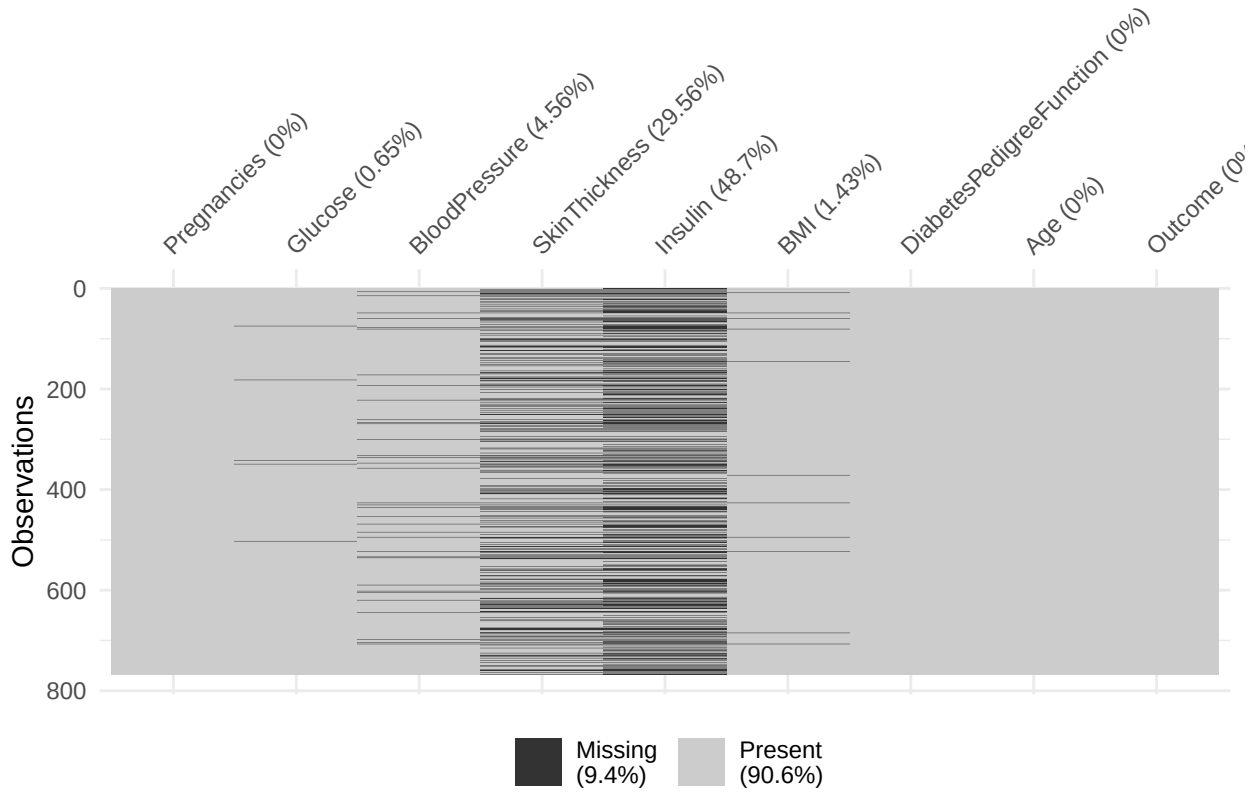
In order for us to later trat them correctly, we need to manually change them to the existing `NA` type. As we do so, we record the number of NaNs instances in each of those variables. For convenience, we define a function.

[1] Remember we are dealing with medical data, not with artificial one. Hence, there are constraints on the values a variable may take.

```r
set_nas <- function(data, fields) {
    percentage <- list()
    for (field in fields) {
        data[[field]][data[[field]] == 0]   <- NA
        percentage[[field]] <- 100 * sum(is.na(data[[field]])) / nrow(data)
    }
    return(list(data = data, percentage = percentage))
}

# Correctly label NaNs
na_fields   <- c("Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI")
data_na     <- set_nas(data, na_fields)
data        <- data_na$data
percentages <- data_na$percentage

# Visualize them
vis_miss(data)
```



Now the next logical step is to impute those NaN values. We do have some concern about the imputation of the "Insulin" variable which is almost half-filled of NaNs. Nevertheless, we decide to impute it. On the followed strategy, we use the "Predictive Mean Matching Imputation" (PMM in short) as it behaves much more robustly than naive mean or median imputations.

```r
pred_mat <- matrix(1, ncol(data), ncol(data), dimnames = list('row' = colnames(data), 'col' = colnames(
pred_mat[,9] <- 0
for (i in 1:8){
```

```
  pred_mat[i,i]<-0
}
data_im <- mice(data, m = 1, method = "pmm", )
```

```
##
##  iter imp variable
##    1   1  Glucose  BloodPressure  SkinThickness  Insulin  BMI
##    2   1  Glucose  BloodPressure  SkinThickness  Insulin  BMI
##    3   1  Glucose  BloodPressure  SkinThickness  Insulin  BMI
##    4   1  Glucose  BloodPressure  SkinThickness  Insulin  BMI
##    5   1  Glucose  BloodPressure  SkinThickness  Insulin  BMI
```

```
data    <- complete(data_im)
```

Visualization of the data

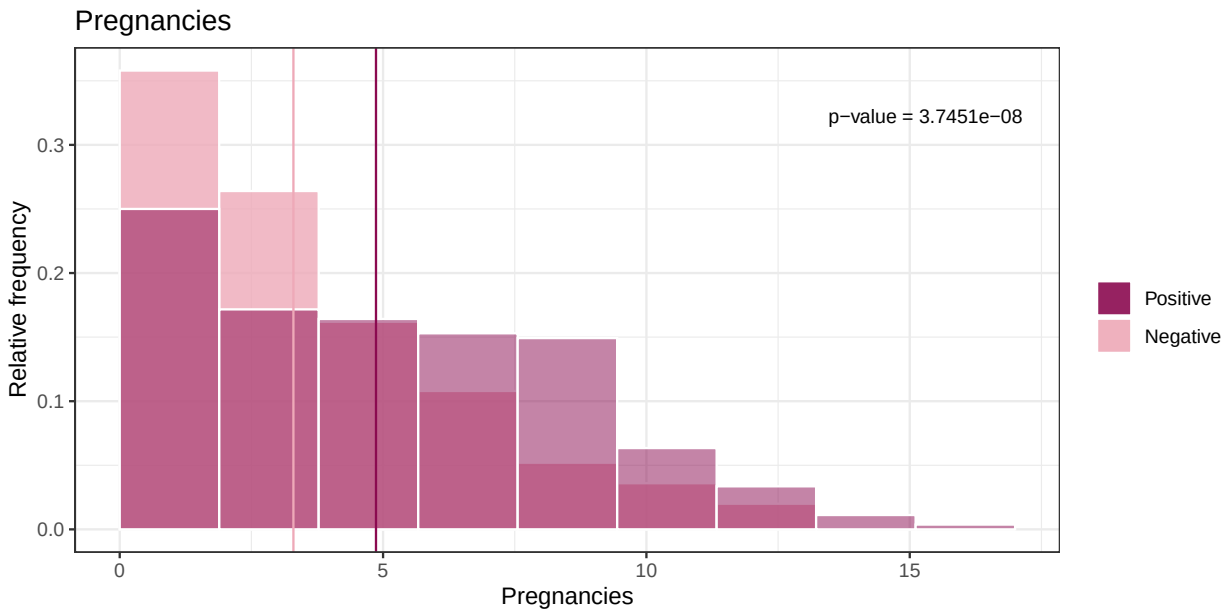
Before we proceed any further, we are going to describe our data. First, we look individually to each of the attributes, we see both visually and with with a two-sample Wilcox-test whether there is significant difference between the two groups (having or not diabetes), and if some transformation may be desirable to ensure normality compatibility. For that purpose, we define the following function.

```
histogram_by_groups <- function(data, var, label = NULL) {
    stat_t <- wilcox.test(as.formula(paste(var, "~ Outcome")), data)
    data0  <- data[data$Outcome == "Negative", ]
    data1  <- data[data$Outcome == "Positive", ]
    if (is.null(label)) {
        label <- var
    }
    p <- ggplot(data0, aes(x = eval(parse(text = var)))) +
        geom_histogram(
            aes(y = after_stat(count / sum(count)), fill = "Negative"),
            bins = 10, colour = "white", alpha = 0.8, boundary = 0
            ) +
        geom_histogram(data = data1,
            aes(
                x = eval(parse(text = var)),
                y = after_stat(count / sum(count)), fill = "Positive"
                ),
            bins = 10, colour = "white",
            alpha = 0.5, boundary = 0, inherit.aes = FALSE) +
            theme_bw() +
            scale_fill_manual(
                name = "",
                breaks = c("Positive", "Negative"),
                values = c("Positive" = "deeppink4", "Negative" = "pink2")
            ) +
            xlab(label) + ylab("Relative frequency") + ggtitle(label) +
            geom_vline(xintercept = mean(data1[[var]]), colour = "deeppink4") +
            geom_vline(xintercept = mean(data0[[var]]), colour = "pink2")
    p + annotate(
            "text",
```

```
            x = 0.9 * max(data1[var]),
            y = 0.9 * max(ggplot_build(p)$data[[1]]["y"]),
            label = sprintf("p-value = %.4e", stat_t$p.value),
            size = 3
        )
}
```

Let us start with the number of pregnancies. We can see that people who have diabetes have had more pregnancies than those who do not have diabetes. We see that it seems to be somewhat based on the p-value alone. We also note it exhibits a heavily right-skewed behaviour. As such, a logarithmic transformation would make sense to get a distribution more compatible with the normal one. Nonetheless, a problem here arises in dealing with the null values [2]. For that purpose we shift the variable by one unit. As a consistency measure, we will apply this shift to all variables we log-transform.

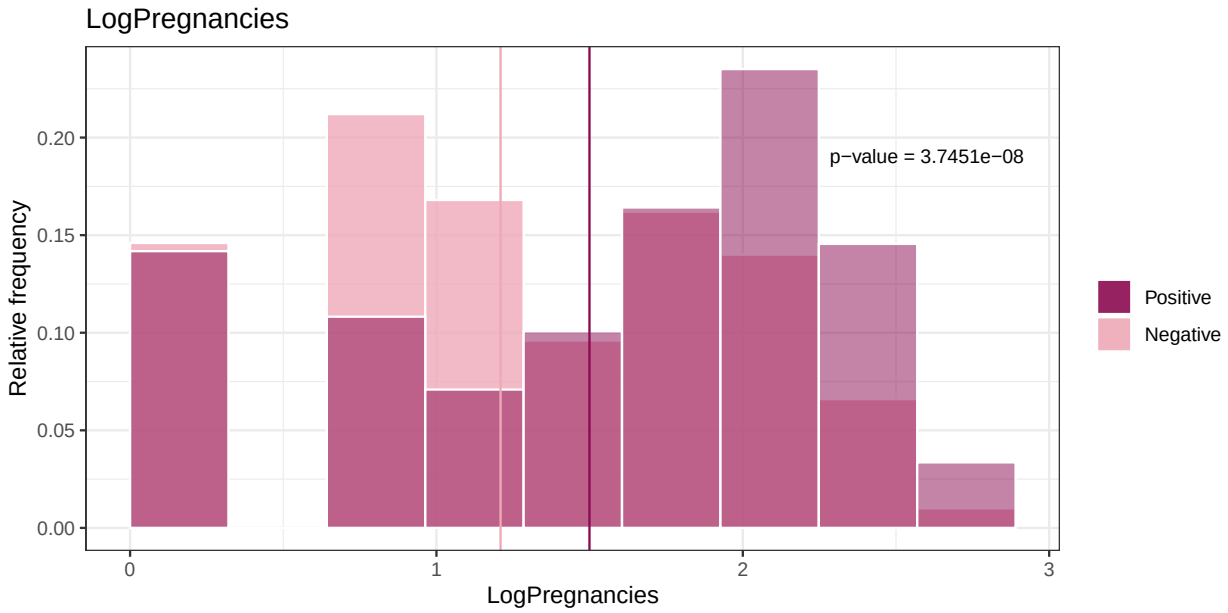
```
histogram_by_groups(data, "Pregnancies")
```



The change as we see does help in obtaining a more centered distribution with which we may better apply the posterior analysis.

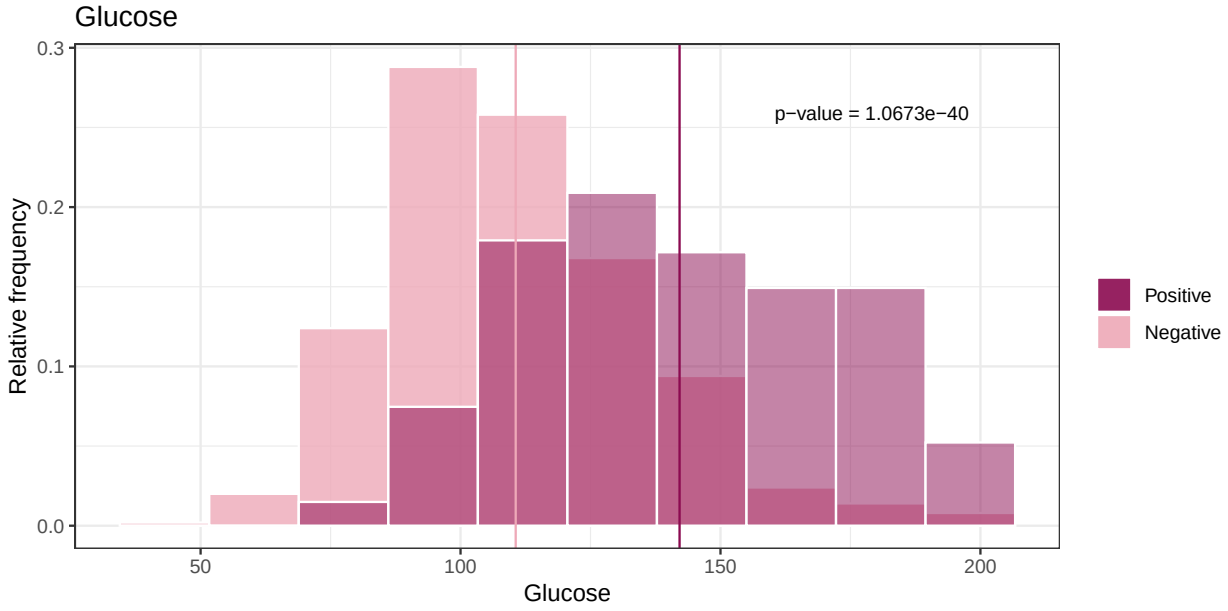
```
data$LogPregnancies <- log(data$Pregnancies + 1)
histogram_by_groups(data, "LogPregnancies")
```

[2] Remember the domain of the logarithmic function is $(0, \infty)$.



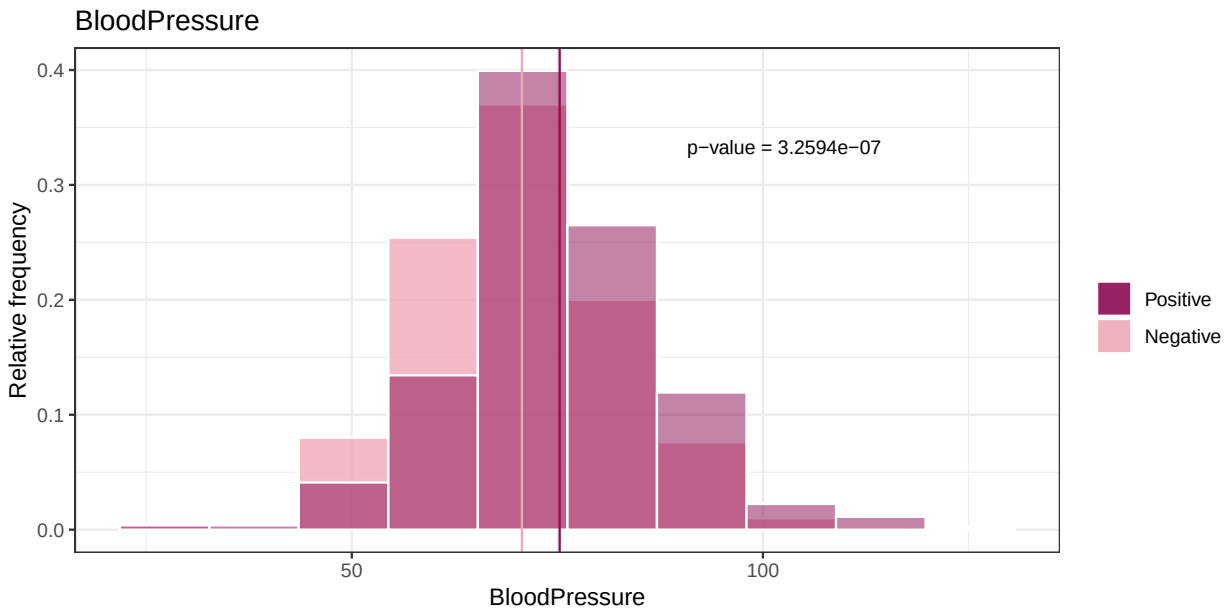
The next variable to visualize is the glucose. People with diabetes exhibit higher glucose values. The p-value is very small which indicates a highly significant difference between the two groups. We also observe that the distribution is already well-centered and resembles a normal distribution. Hence, we decide not to transform the data.

```
histogram_by_groups(data, "Glucose")
```



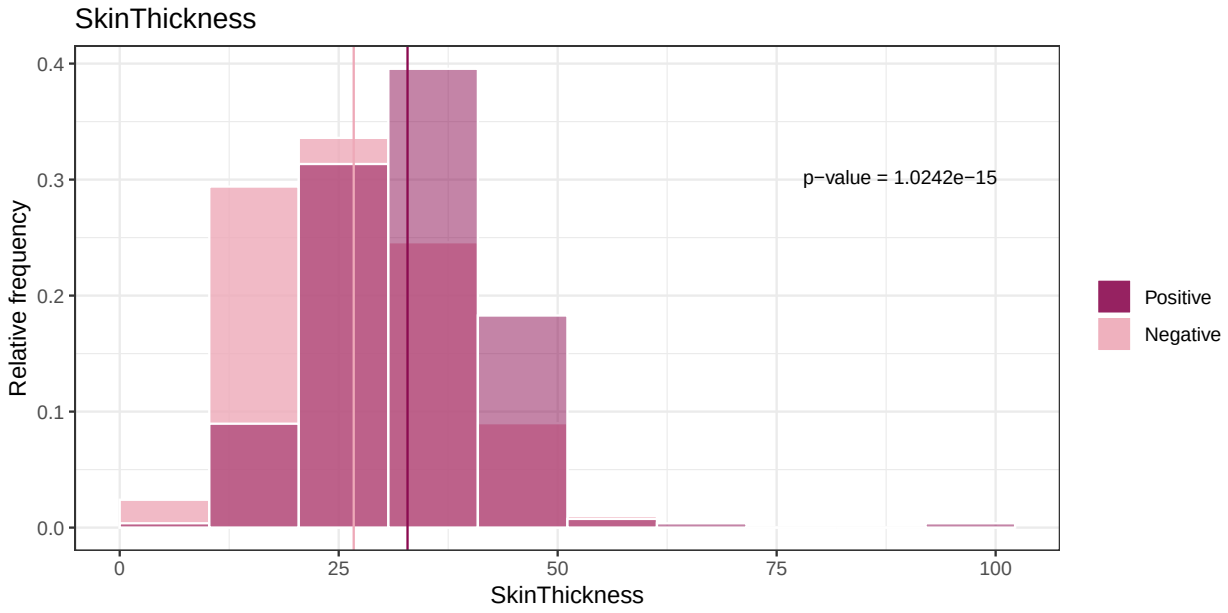
We move onwards to blood pressure. In this case, although the distributions appear as normal, there does not seem to exist a highly significant difference between the groups in contrast to what the p-value states, more so compared with the glucose variable. Nonetheless, there seem to be an slight indication of higher blood pressure for people with diabetes.

```
histogram_by_groups(data, "BloodPressure")
```



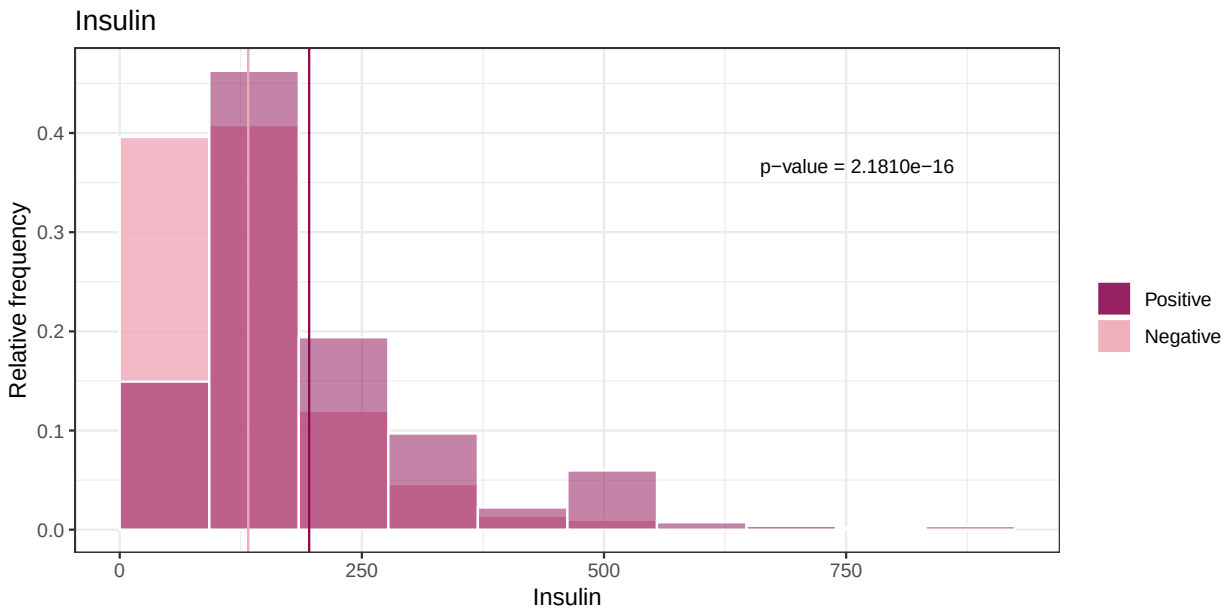
Now skin thickness is interesting, because the distributions are visually to those of the blood pressure but the presence of outliers is appreciable. We will later deal with those but for now let us mantain this variable as it is. Note that the population with diabetes appear to exhibit a thicker skin.

```
histogram_by_groups(data, "SkinThickness")
```



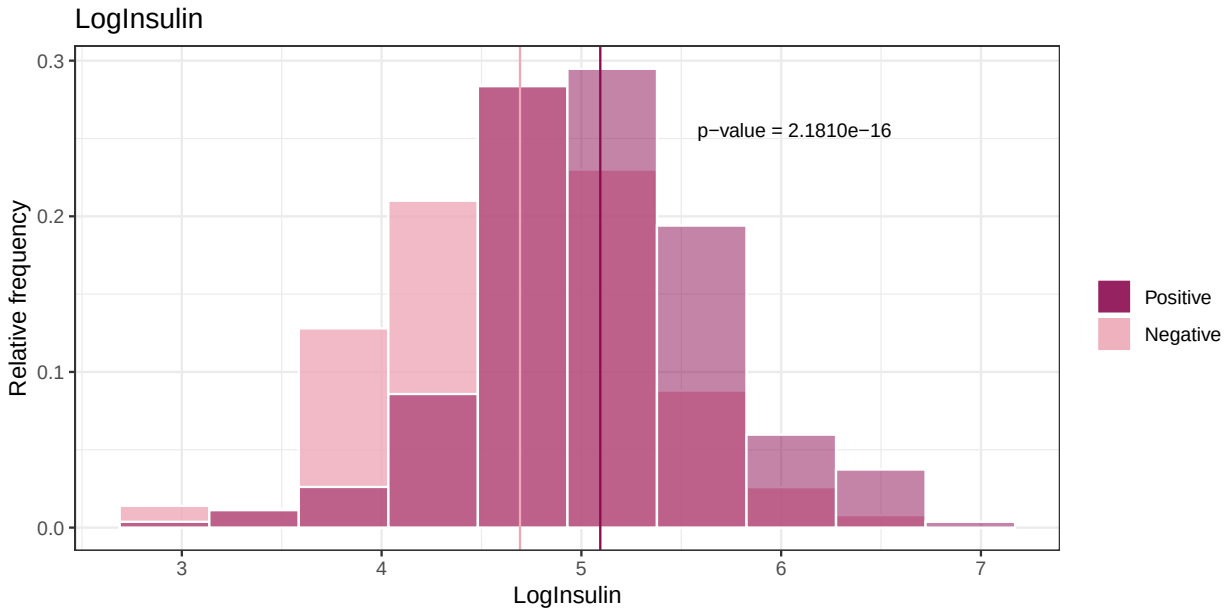
Insulin, as we may have expected from the name of the property itself, appears to be a relevant. The median of the distributions does indeed seem to differ, with the one for the diabetes population being slightly higher. It is also right-skewed, so we decide to log-transform it.

```
histogram_by_groups(data, "Insulin")
```



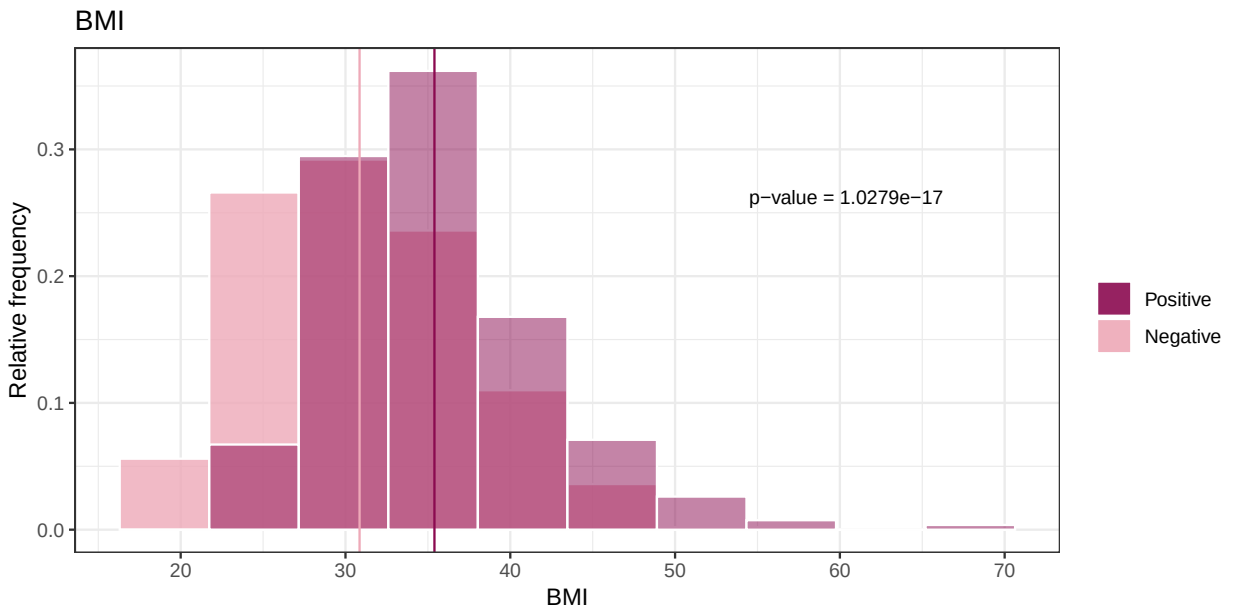
It does appear that the transformation impproves the the symmetrization of the data, although some left-skewness appears. We will later see if outlier detection get to target those values or not.

```
data$LogInsulin <- log(data$Insulin + 1)
histogram_by_groups(data, "LogInsulin")
```



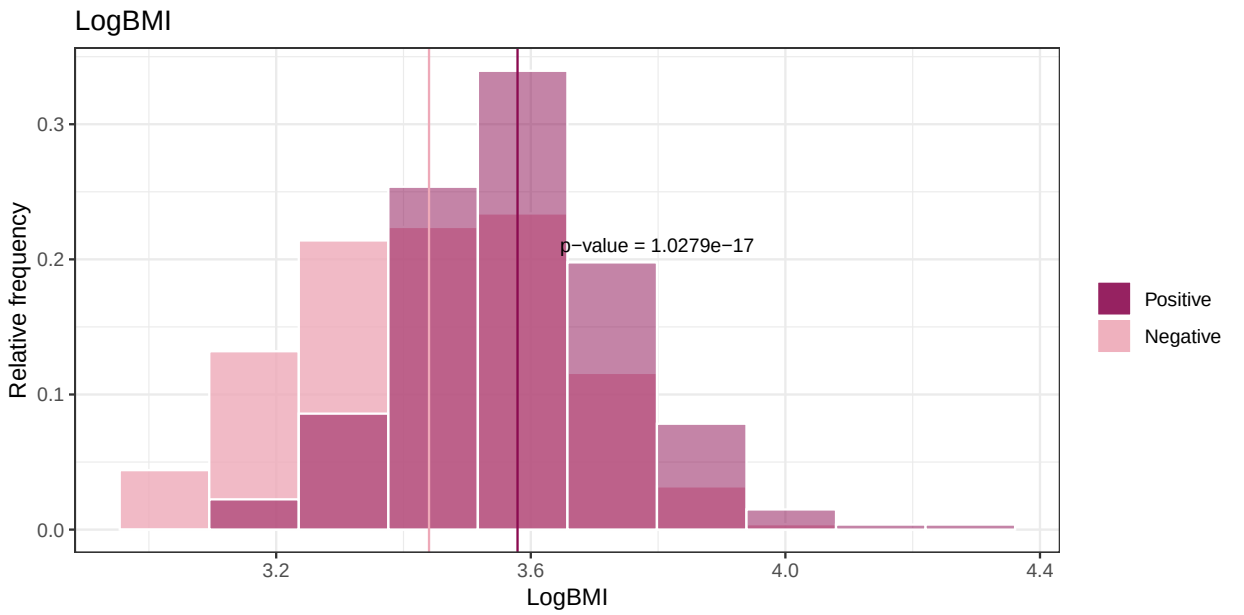
Body Mass Index (BMI) again exhibits this tendency of leaning towards a more right-skewed distribution. It does also follow the tendency of being slightly higher for people with diabetes. As such we log-transform to try and get a more normalized variable.

```
histogram_by_groups(data, "BMI")
```



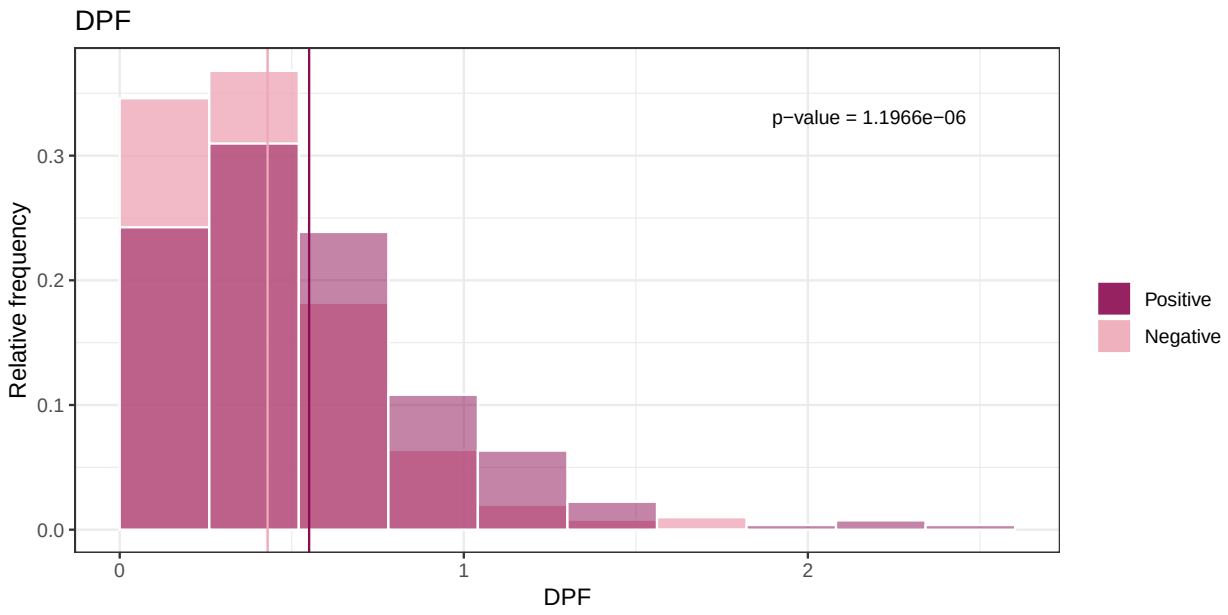
As we may visually judge, it is the case that the log transformation centers the data and provides a more normal-like distribution. There is some presence of seemingly outlying points to the right.

```
data$LogBMI <- log(data$BMI + 1)
histogram_by_groups(data, "LogBMI")
```



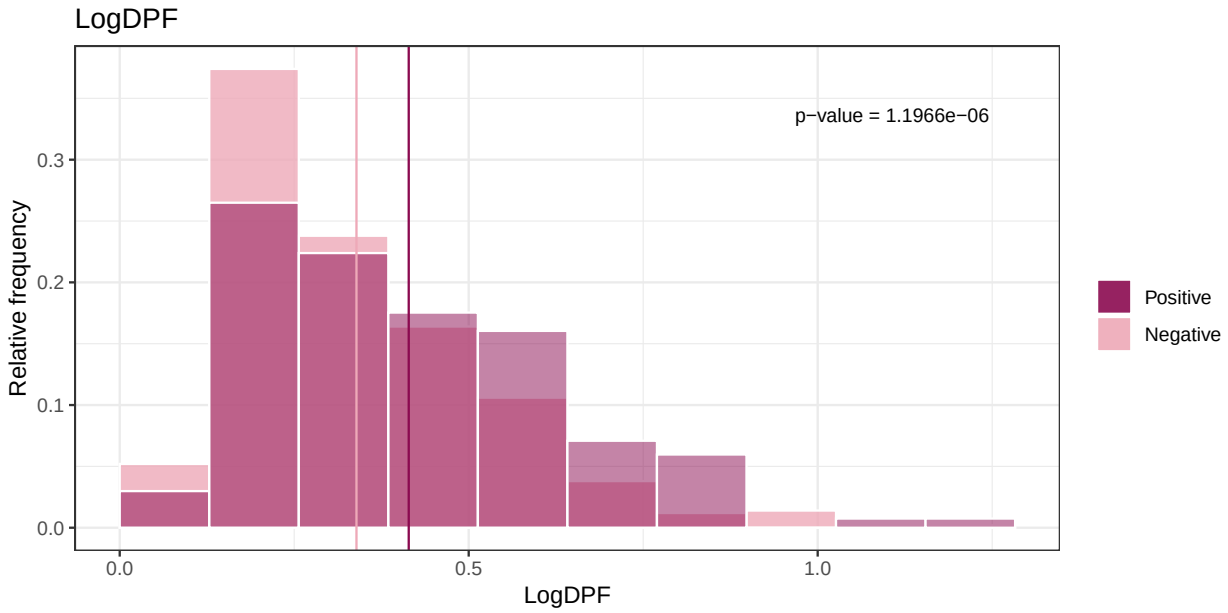
The Diabetes Pedigree Function (DPF) is the again a flagrant right-skewed. It again has higher values for the positive set. This is to be expected from the definition of this very function as a risk indication for diabetes. Let us try to log-transform it.

```
histogram_by_groups(data, "DiabetesPedigreeFunction", "DPF")
```



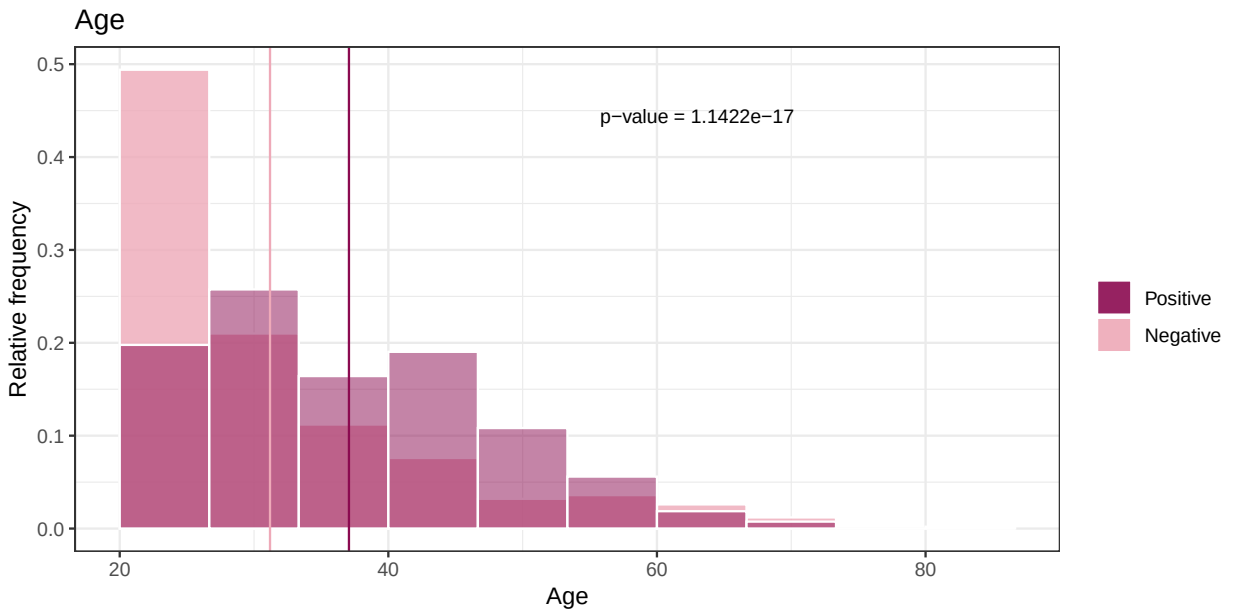
The improvement is highly noticeable. We retain thus this transformed variable.

```
data$LogDPF <- log(data$DiabetesPedigreeFunction + 1)
histogram_by_groups(data, "LogDPF")
```



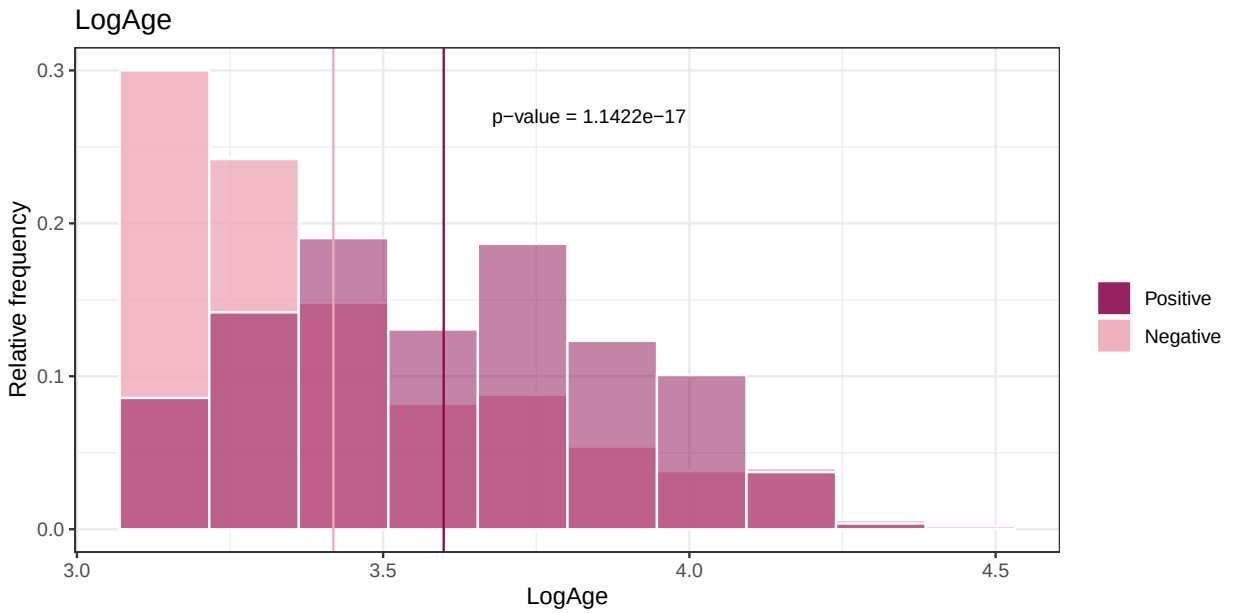
We may note that young people, as with other illnesses have less tendency to suffer diabetes than the elders. The age is expected to exhibit a right-skewed distribution, as is indeed the case. In an attempt to improve the symmetry, we once again use logarithms to transform the variable.

```
histogram_by_groups(data, "Age")
```



The transformation does help although not by much. This is a common problem of the age variable. We keep the transformation anyway as it does seem to help with the symmetry for the positive group.

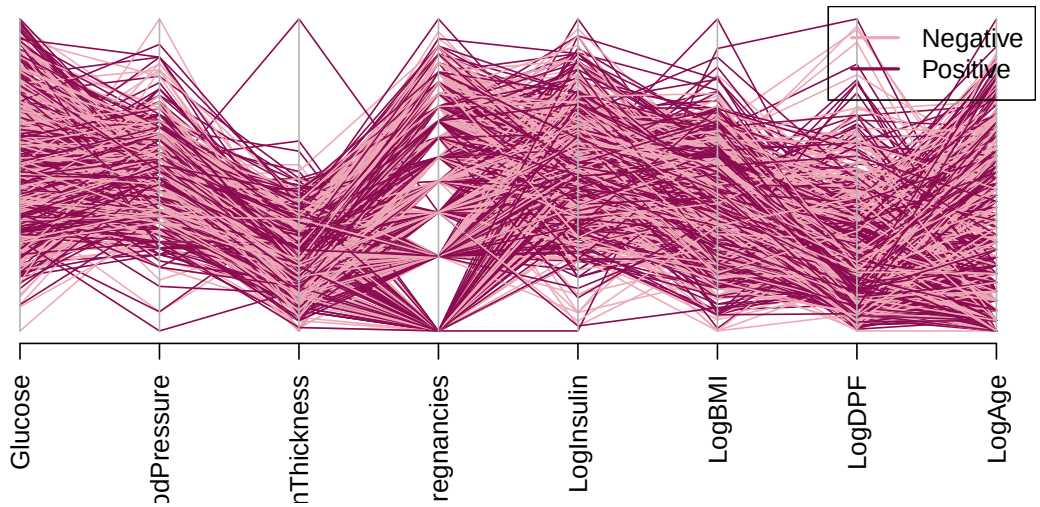
```
data$LogAge <- log(data$Age + 1)
histogram_by_groups(data, "LogAge")
```




```
# Some convenience
df  <- subset(data, select = -c(Pregnancies, Insulin, BMI, DiabetesPedigreeFunction, Age))
df0     <-  df[df$Outcome == "Negative", ]
df1     <-  df[df$Outcome == "Positive", ]
xnames  <- names(df)[! names(df) %in% c("Outcome")]
```

Now, let's take a look at some multivariate plots. We'll begin by inspecting the Parallel Coordinate Plot:

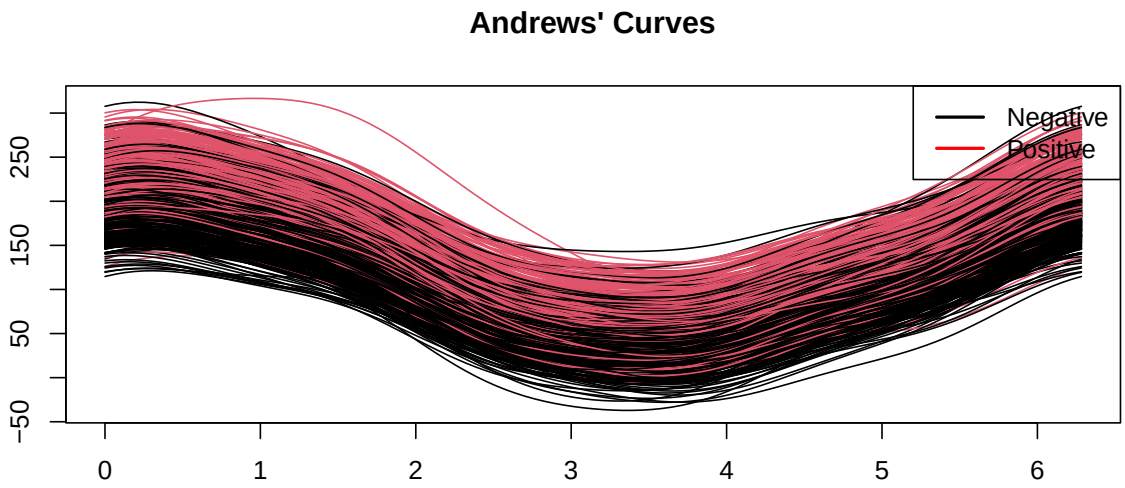
```
par(las = 2)
parcoord(df[xnames], col = c("pink2", "deeppink4"))
legend("topright", legend = c("Negative", "Positive"),
       col = c("pink2", "deeppink4"), lty = 1, lwd = 2)
```



It seems that, overall, the positive lines are over negative ones. This is most notable on the Glucose and log transformed BMI. They are two of the most significant features according to the p-value.

The Andrew's plot is the following:

```
andrewsplot(as.matrix(df[xnames]), df$Outcome, style = "cart")
legend("topright", legend = c("Negative", "Positive"),
       col = c("black", "red"), lty = 1, lwd = 2)
```



Again, we see that the two groups are different. The group of people who have diabetes tend to have more volatile curves, reaching higher and lower values along the curve.

Multivariate characteristics and outlier identification

We are goig to use a multivariate approach to identifying the outliers in our data. In the process, we will need to compute the mean and covariance.

We start then by finding the mean vector and the covariance matrix. In order to reduce the sensitivity to outliers in this computation, we use a robust estimation using the "Fast MCD" (Minimum Covariance Determinant) estimator. We set main parameter, `alpha`, which determines the percentage of the data to use, at 0.85.

```
our_corrplot <- function(cov_mat) {
    colnames(cov_mat) <- c("Log\nPregnancies",
                           "Glucose",
                           "Blood\nPressure",
                           "Skin\nThickness",
                           "LogInsulin",
                           "LogBMI",
                           "LogDPF",
                           "LogAge")
    corrplot.mixed(cov2cor(cov_mat), lower = "number", upper = "color",
        diag = "n", tl.col = "black", tl.cex = 0.65,
        lower.col = "black")
}

mcd_est <- CovMcd(df[xnames], alpha = 0.85, nsamp = "deterministic")
```

```
## [1] "The mean vector is:"
```

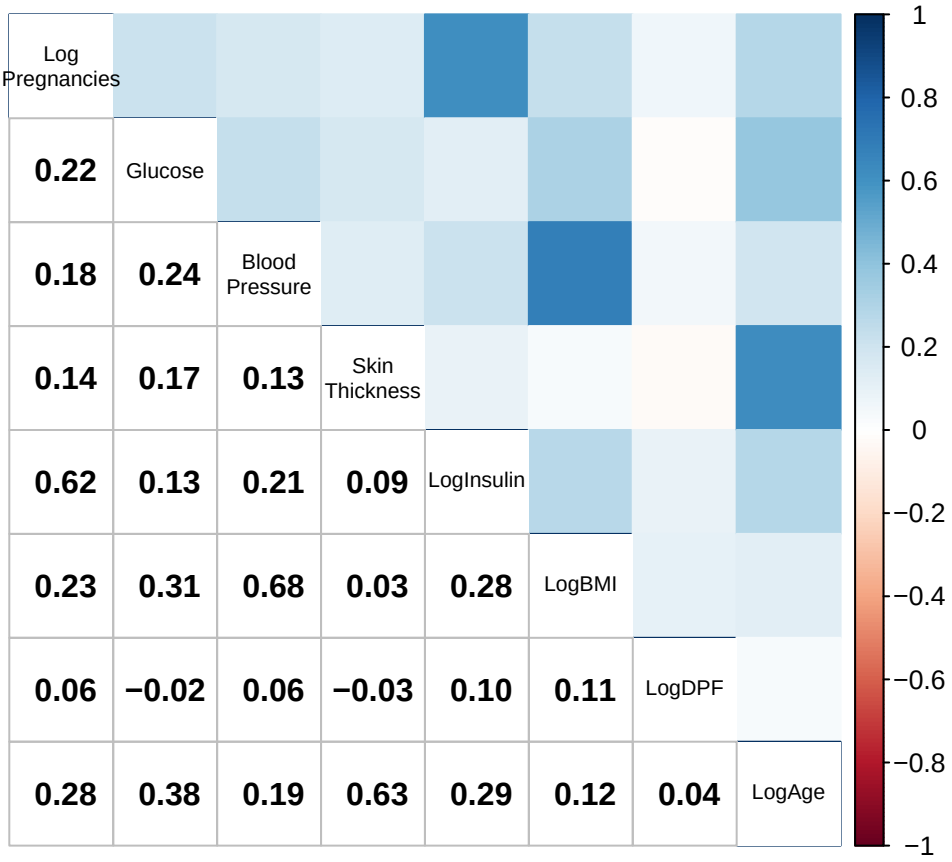
```
##         Glucose   BloodPressure   SkinThickness LogPregnancies      LogInsulin
##     120.2384937      71.7601116      28.4797768       1.3343529       4.8256948
```

```
##          LogBMI         LogDPF         LogAge
##       3.4833183      0.3525359      3.4653539
```

```
## [1] "The covariance matrix is:"
```

```
##                 Glucose BloodPressure SkinThickness LogPregnancies LogInsulin
## Glucose        930.3524       77.2039       56.1109          3.4364    12.3940
## BloodPressure   77.2039      137.7727       28.2458          1.6132     0.9813
## SkinThickness   56.1109       28.2458      104.7185          1.0600     1.4382
## LogPregnancies   3.4364        1.6132        1.0600          0.6188     0.0483
## LogInsulin      12.3940        0.9813        1.4382          0.0483     0.4354
## LogBMI           1.4536        0.7608        1.4375          0.0052     0.0375
## LogDPF           0.3508       -0.0386        0.1123         -0.0041     0.0122
## LogAge           2.7443        1.4369        0.6233          0.1566     0.0607
##                  LogBMI  LogDPF LogAge
## Glucose          1.4536  0.3508 2.7443
## BloodPressure    0.7608 -0.0386 1.4369
## SkinThickness    1.4375  0.1123 0.6233
## LogPregnancies   0.0052 -0.0041 0.1566
## LogInsulin       0.0375  0.0122 0.0607
## LogBMI           0.0426  0.0043 0.0082
## LogDPF           0.0043  0.0365 0.0022
## LogAge           0.0082  0.0022 0.1015
```

```
our_corrplot(mcd_est$cov)
```



It is interesting to seggregate by the class of the Outcome variable. This way, we can both, get the mean vector and covariance matrix for each of the classes, and the outliers for both sets.

```
mcd_neg <- CovMcd(df0[xnames], alpha = 0.85, nsamp = "deterministic")
mcd_pos <- CovMcd(df1[xnames], alpha = 0.85, nsamp = "deterministic")
```

```
## [1] "### Negative class ###"
```

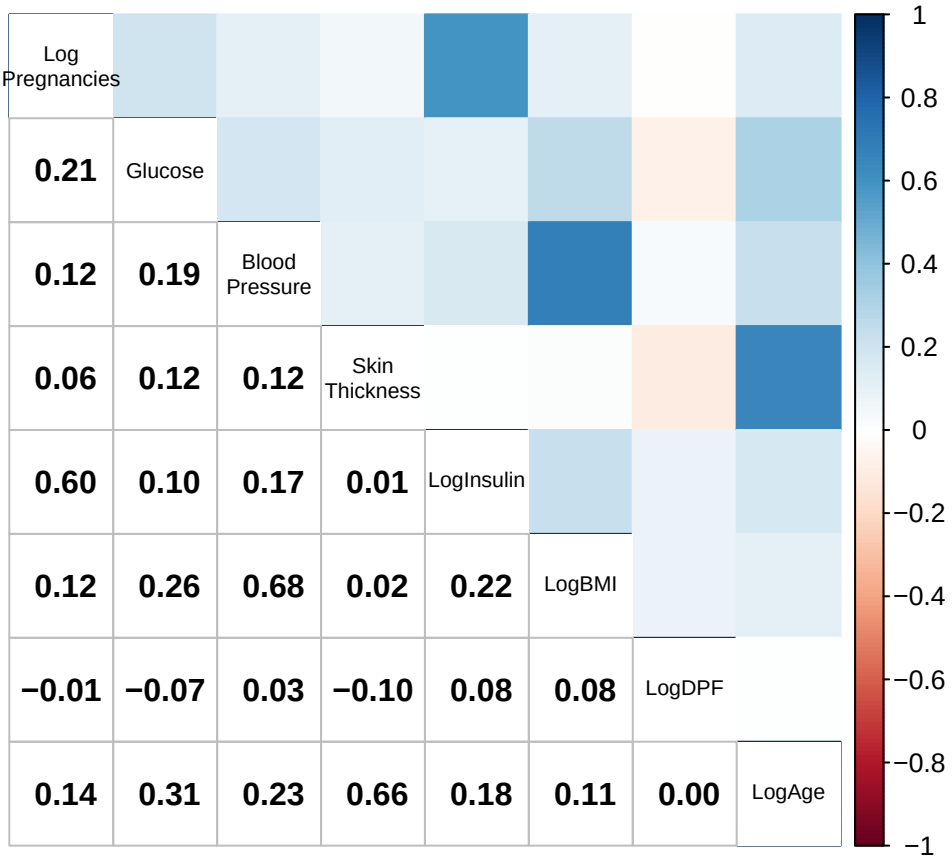
```
## [1] "The mean vector is:"
```

```
##        Glucose  BloodPressure  SkinThickness LogPregnancies     LogInsulin
##     108.629386      69.969298      26.701754       1.212348       4.669472
##         LogBMI         LogDPF         LogAge
##       3.438084       0.323836       3.382709
```

```
## [1] "The covariance matrix is:"
```

```
##                 Glucose BloodPressure SkinThickness LogPregnancies LogInsulin
## Glucose        533.9858       55.2003       28.6707         0.9896     8.9124
## BloodPressure   55.2003      134.1355       23.4175         1.0585     0.7860
## SkinThickness   28.6707       23.4175      113.4519         0.9351     1.1567
## LogPregnancies   0.9896        1.0585        0.9351         0.5690     0.0028
## LogInsulin       8.9124        0.7860        1.1567         0.0028     0.4185
## LogBMI           0.5820        0.6303        1.5452         0.0029     0.0305
## LogDPF          -0.0298       -0.1402        0.0561        -0.0132     0.0090
## LogAge           0.9620        1.0743        0.7074         0.1463     0.0339
##                LogBMI  LogDPF LogAge
## Glucose        0.5820 -0.0298 0.9620
## BloodPressure  0.6303 -0.1402 1.0743
## SkinThickness  1.5452  0.0561 0.7074
## LogPregnancies 0.0029 -0.0132 0.1463
## LogInsulin     0.0305  0.0090 0.0339
## LogBMI         0.0451  0.0030 0.0070
## LogDPF         0.0030  0.0294 0.0000
## LogAge         0.0070  0.0000 0.0867
```

```
our_corrplot(mcd_neg$cov)
```




```
## [1] "### Positive class ###"
```

```
## [1] "The mean vector is:"
```

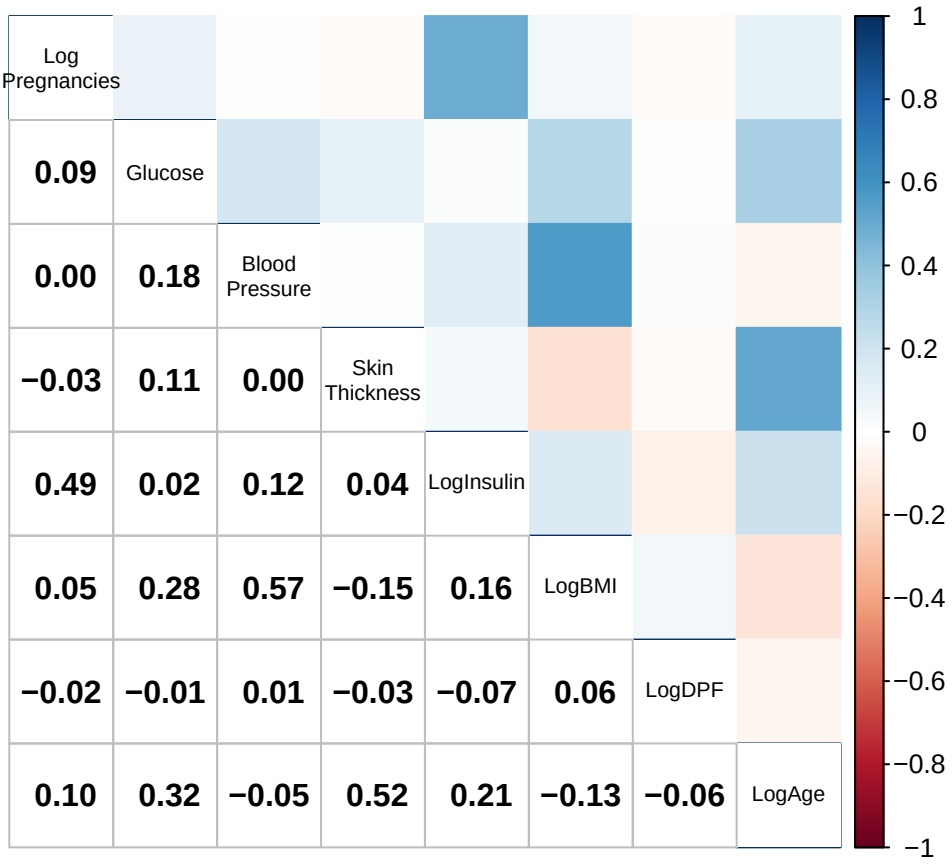
```
##        Glucose  BloodPressure  SkinThickness LogPregnancies     LogInsulin
##     141.945312      75.515625      32.378906       1.525330       5.102782
##         LogBMI         LogDPF         LogAge
##       3.574339       0.400851       3.596905
```

```
## [1] "The covariance matrix is:"
```

```
##                 Glucose BloodPressure SkinThickness LogPregnancies LogInsulin
## Glucose        925.0235       31.0959       -0.8711        -0.7283     9.0825
## BloodPressure   31.0959      143.4356       19.2008         1.0827     0.1324
## SkinThickness   -0.8711       19.2008       76.9575         0.0170     0.6476
## LogPregnancies  -0.7283        1.0827        0.0170         0.6813     0.0212
## LogInsulin       9.0825        0.1324        0.6476         0.0212     0.3651
## LogBMI           0.2789        0.5837        0.8491        -0.0215     0.0163
## LogDPF          -0.1256       -0.0196        0.0203        -0.0045    -0.0088
## LogAge           0.9015        1.1281       -0.1330         0.1249     0.0380
##                 LogBMI  LogDPF  LogAge
## Glucose         0.2789 -0.1256  0.9015
## BloodPressure   0.5837 -0.0196  1.1281
## SkinThickness   0.8491  0.0203 -0.1330
## LogPregnancies -0.0215 -0.0045  0.1249
```

```
## LogInsulin      0.0163 -0.0088  0.0380
## LogBMI          0.0293  0.0021 -0.0067
## LogDPF          0.0021  0.0421 -0.0036
## LogAge         -0.0067 -0.0036  0.0857
```

```
our_corrplot(mcd_pos$cov)
```



Let us focus first on the mean vectors. The here observed particularities are nothing new, as they were already present on above histograms:

- Higher number of pregnancies seems to increase the chances on developing diabetes.
- The glucose and insulin levels of the positive population is higher in contrast to the negative one.

Looking now at the representations for the covariance matrices, the major changes are that the correlation between skin thickness and diabetes pedigree function is lower in the group who do not have diabetes than in the one having the disease. Moreover, the correlation between BMI [3] and age is positive for the sane group while negative on the positive one.

We now search for outliers. The idea is to use Mahalanobis distance. As we suppose that our data $X \sim \mathcal{N}(\mu, \Sigma)$, we have $D_M(x, \mu)^2 \sim \chi^2_p$, with D_M the Mahalanobis distance. Hence we may set our outlier criteria as

$$D_M(x, \mu)^2 > \chi^2_{p,0.95^{1/n}} \, , \tag{1}$$

the 0.95^{th} quantile of the χ^2_p distribution. We then drop these outliers.

[3] Remember this is the logarithm.

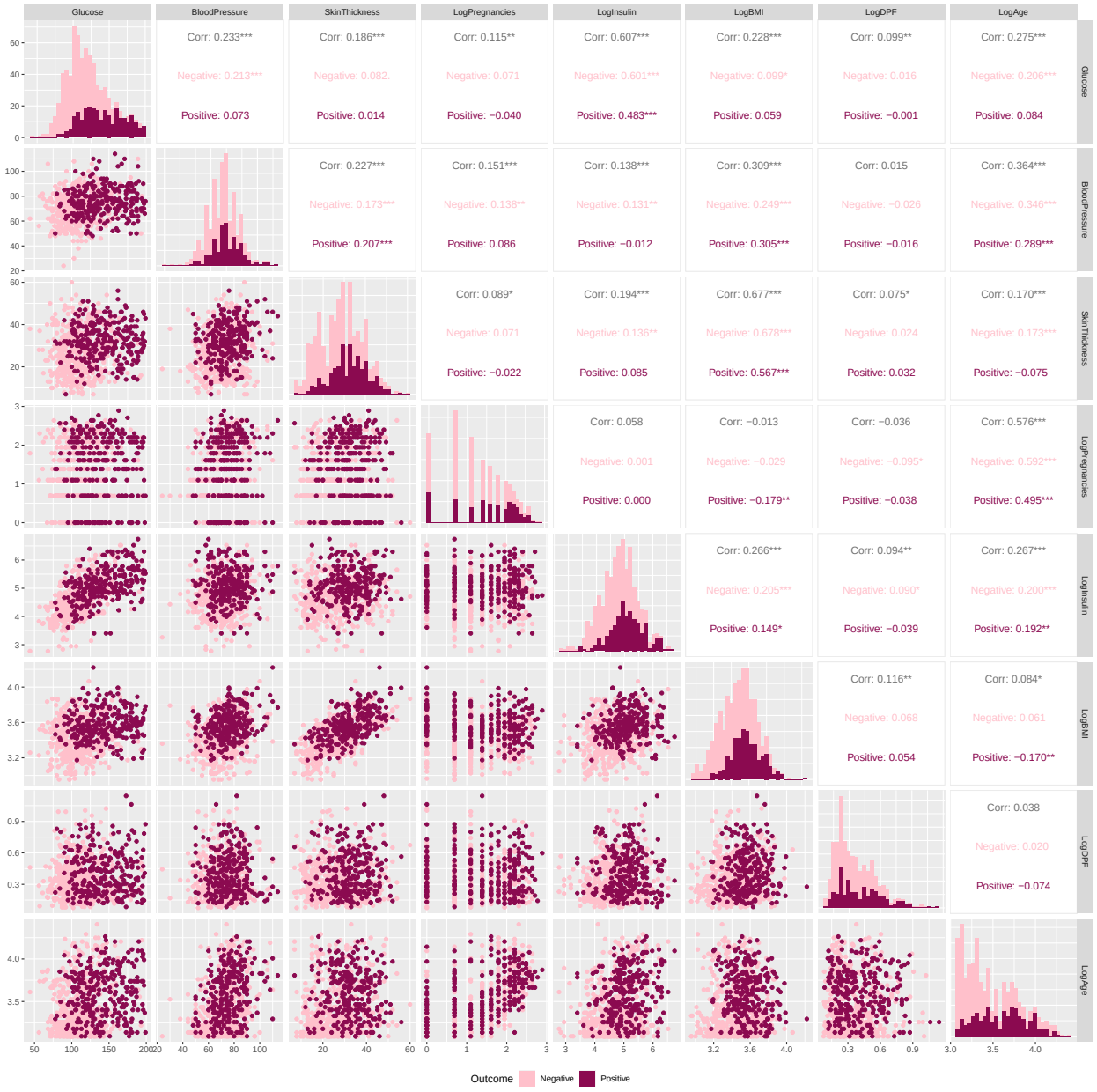
```
p   <- length(xnames)
n0  <- nrow(df0)
n1  <- nrow(df1)
df0_clean <- df0[mcd_neg$mah < qchisq(0.95^(1 / n0), p), ]
df1_clean <- df1[mcd_pos$mah < qchisq(0.95^(1 / n1), p), ]
df_clean  <- rbind(df0_clean, df1_clean)
```

```
## [1] "The negative set contained 6 outliers."
```

```
## [1] "The positive set contained 4 outliers."
```

As a final summary of this section, we perform a plot in which the histograms of different populations are observed, as well as scatterplots of pairs of variables and correlations.

```
ggpairs(df_clean, aes(color = Outcome), legend = 1,
        columns = xnames,
        diag = list(continuous = "barDiag")
        ) +
    theme(legend.position = "bottom") +
    scale_fill_manual(values = c("pink", "deeppink4")) +
    scale_color_manual(values = c("pink", "deeppink4")) + labs(fill = "Outcome")
```



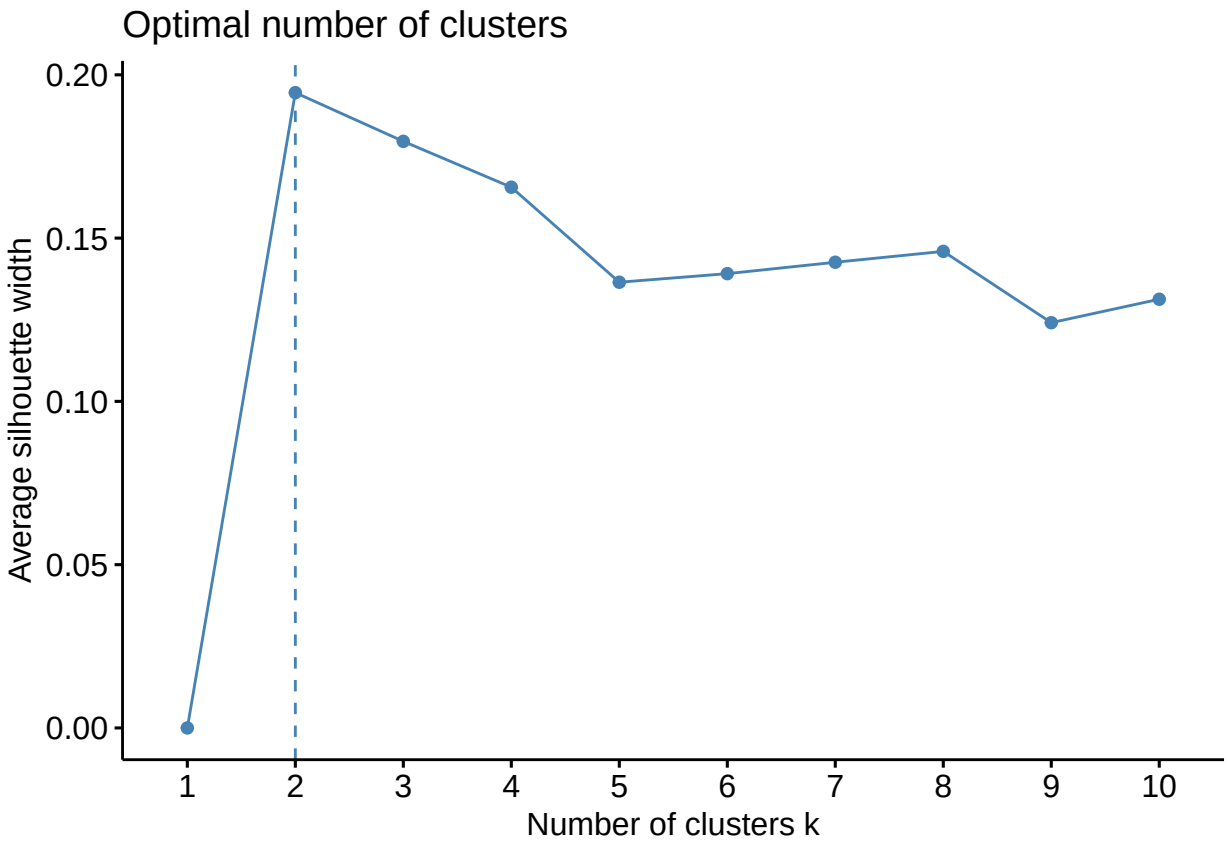
Unsupervised learning

Partitional clustering

Number of clusters

We'll firstly decide how many clusters we want to create. We'll use the silhoutte method and the gap statistic. The silhoutte method:

```
fviz_nbclust(X2, kmeans, method = "silhouette", k.max = 10)
```

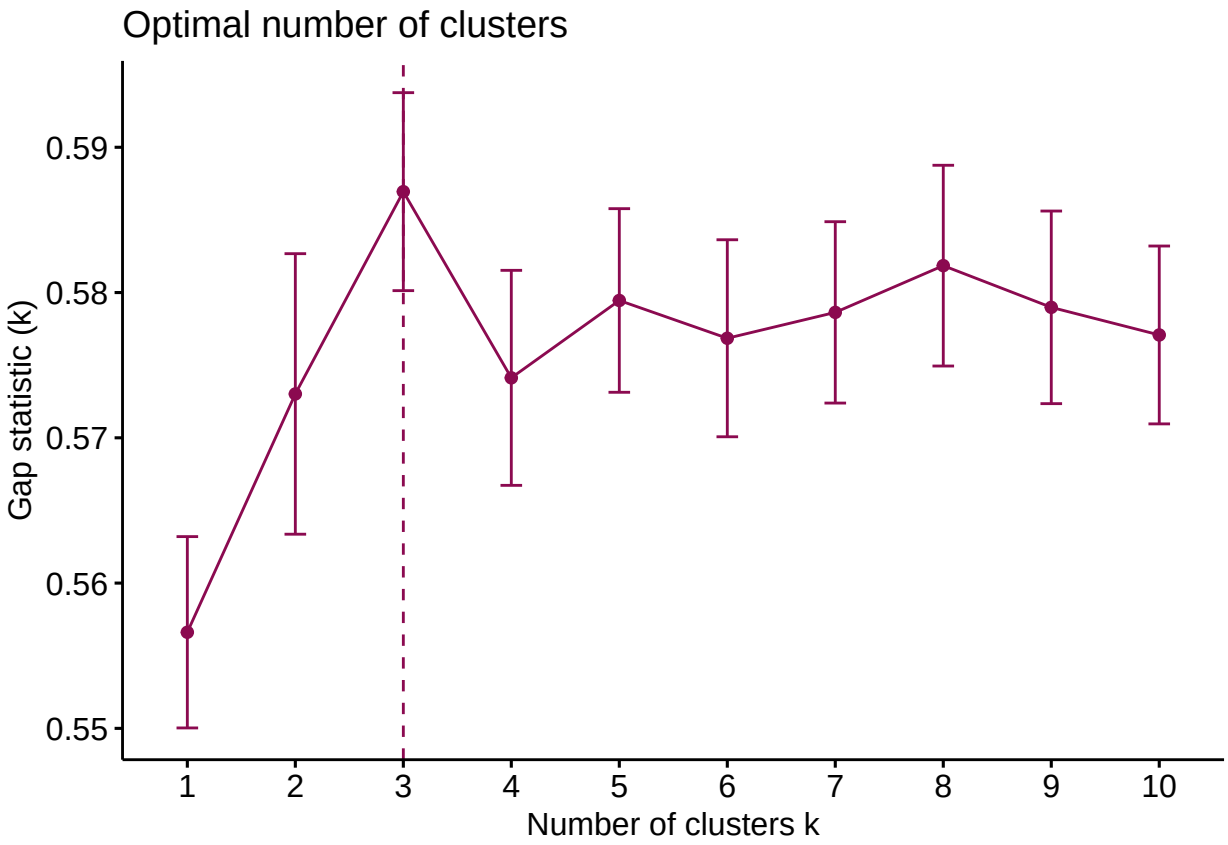


As we can see, this indicates that the optimum number of clusters is 2.

The gap statistic:

```
gap_stat <- clusGap(X2, FUN = kmeans, K.max = 10, B = 100)
```

```
fviz_gap_stat(gap_stat, linecolor = c1, maxSE =
                list(method = "firstSEmax", SE.factor = 1))
```



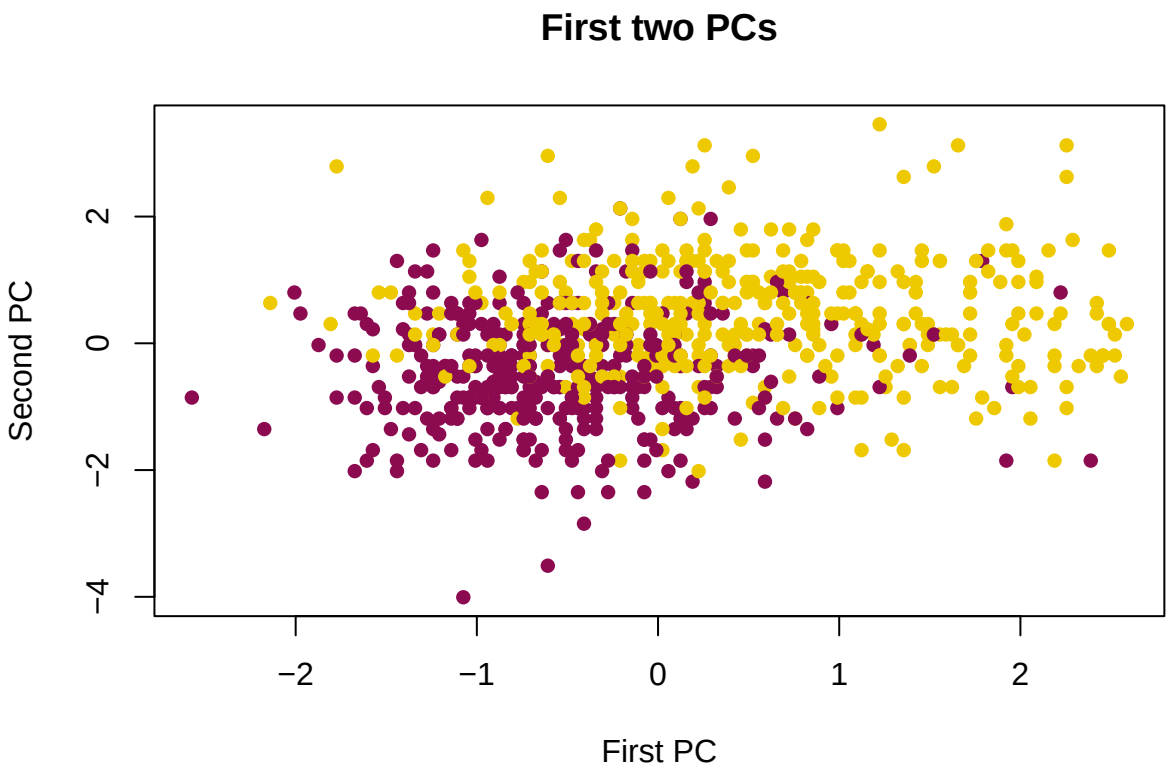
This also suggests that the optimal number of clusters is 2.

K-Means

The results of k-means clustering are the following:

```
kmeans_X <- kmeans(X2, centers = 2, iter.max = 1000, nstart = 100)

kmeans_col <- c(c1, c2)[kmeans_X$cluster]
plot(X2[,1:2], col = kmeans_col, main = "First two PCs",
     xlab = "First PC", ylab = "Second PC", pch  = 16)
```



Let's see the average in each variable for the clusters:

```
kmeans_means <- cluster_means(kmeans_X$cluster)
kable(kmeans_means)
```

	Cluster 1	Cluster 2
Glucose	105.092896	136.3801020
BloodPressure	66.633880	77.6454082
SkinThickness	23.412568	33.6301020
LogPregnancies	1.014614	1.6050048
LogInsulin	4.495434	5.1442477
LogBMI	3.387851	3.5796160
LogDPF	0.338591	0.3814392
LogAge	3.284747	3.6589152

We can also see the confusion matrix with our target variable, although we must remember that this is not the objective of unsupervised learning.

```
kable(cluster_classification(kmeans_X$cluster, first = 1))
```

	Negative	Positive
Negative	318	48
Positive	176	216

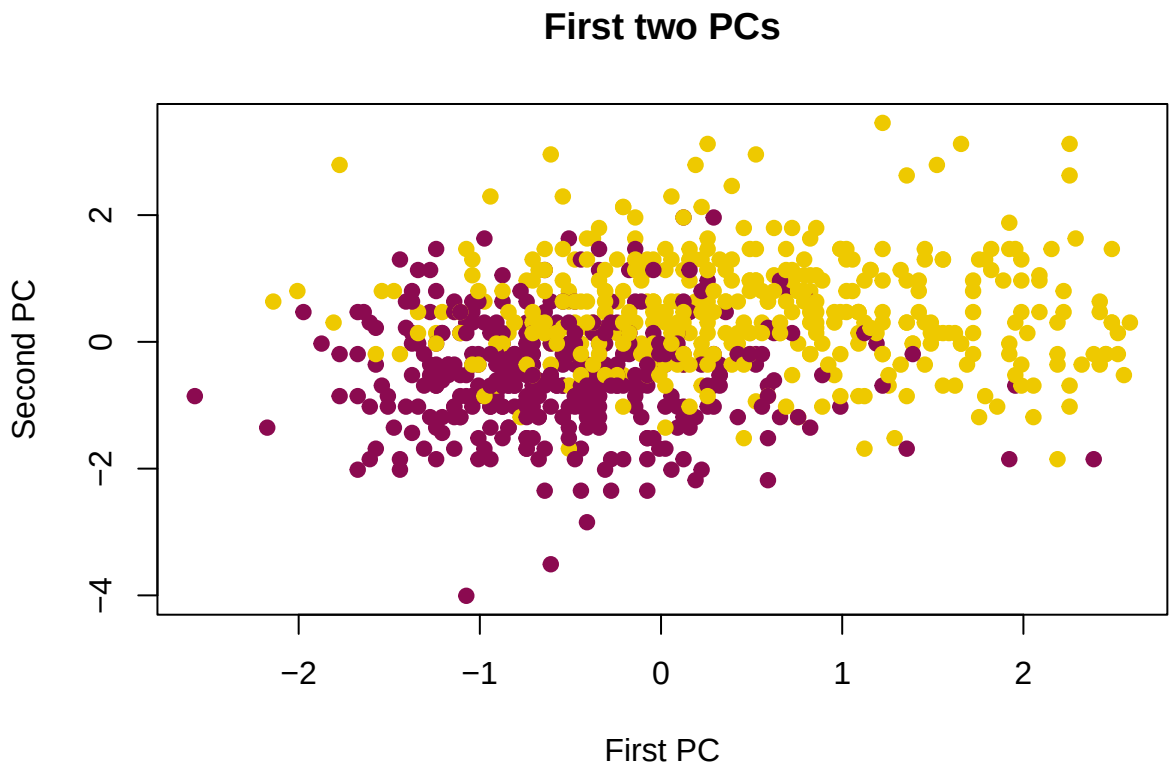
As we can see, the clusters are not exactly coincident with the target variable.

K-Medoids

As all of our attributes are numerical and we don't have any reason not to, we will use Euclidean distance.

```
pam_X <- pam(X2, k = 2, metric = "euclidean", stand = FALSE)
pam_col <- c(c1, c2)[pam_X$cluster]

plot(X2[,1:2], col = pam_col, main = "First two PCs",
     xlab = "First PC", ylab = "Second PC", pch = 19)
```



We obtain a very similar result than with K-Means.

```
pam_means <- cluster_means(pam_X$cluster, first = 1)
kable(pam_means)
```

	Cluster 1	Cluster 2
Glucose	104.9078212	135.9200000
BloodPressure	65.9189944	78.0650000
SkinThickness	24.2262570	32.6975000
LogPregnancies	1.0047290	1.6020445
LogInsulin	4.4895534	5.1365345
LogBMI	3.4004414	3.5645122
LogDPF	0.3405789	0.3788031
LogAge	3.2620573	3.6717387

Again, the confusion matrix is the following:

```
cluster_classification(pam_X$cluster)
```

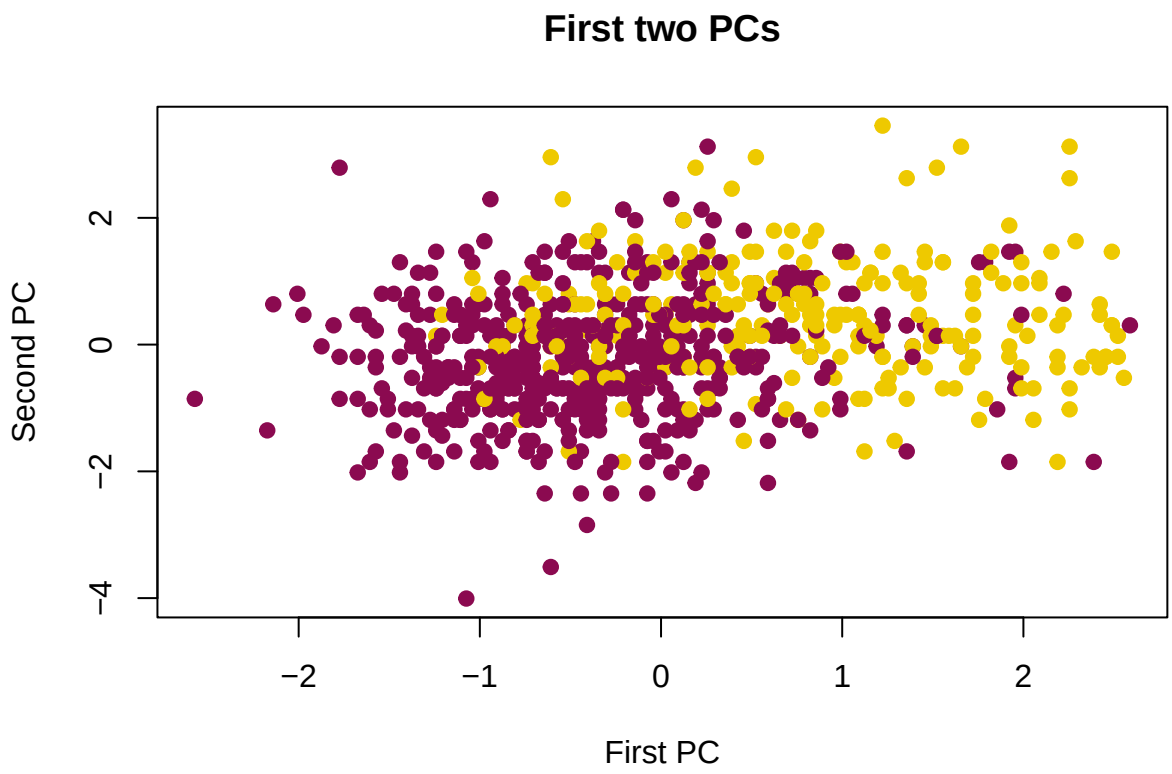
```
##           Actual
## Pred       Negative Positive
##   Negative      309       49
##   Positive      185      215
```

Very similar results than with K-Means.

CLARA

```
clara_X <- clara(X2, k = 2, metric = "euclidean", stand=FALSE)
clara_col <- c(c1, c2)[clara_X$cluster]

plot(X2[,1:2], col = clara_col, main = "First two PCs",
     xlab = "First PC", ylab = "Second PC", pch = 19)
```



Again, we can observe that the results are very similar than with both K-Means and K-Medoids.

```
clara_means <- cluster_means(clara_X$cluster, first = 1)
kable(clara_means)
```

	Cluster 1	Cluster 2
Glucose	110.2158416	143.3438735
BloodPressure	69.7702970	77.4347826
SkinThickness	27.1584158	31.7667984
LogPregnancies	1.0358758	1.8869308
LogInsulin	4.6035272	5.2849513
LogBMI	3.4580310	3.5448903
LogDPF	0.3675311	0.3472144
LogAge	3.3237338	3.7866658

```
cluster_classification(clara_X$cluster)
```

```
##           Actual
## Pred       Negative Positive
##   Negative      391      114
##   Positive      103      150
```

Hierarchical clustering

Single linkage

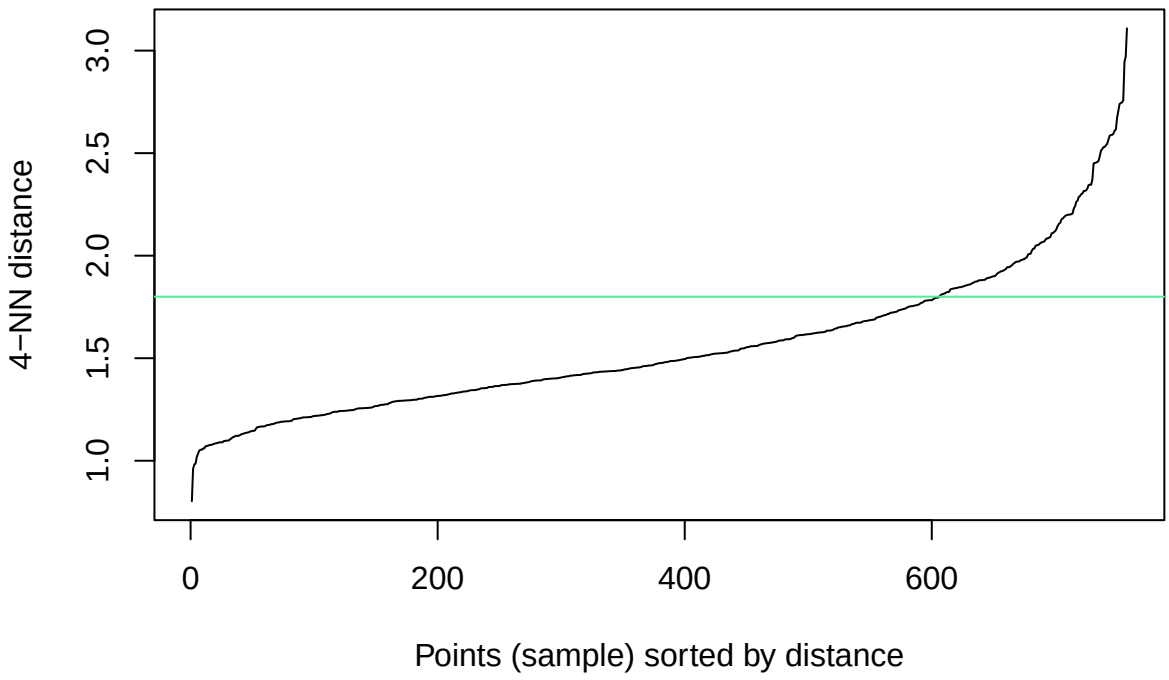
Complete linkage

Average linkage

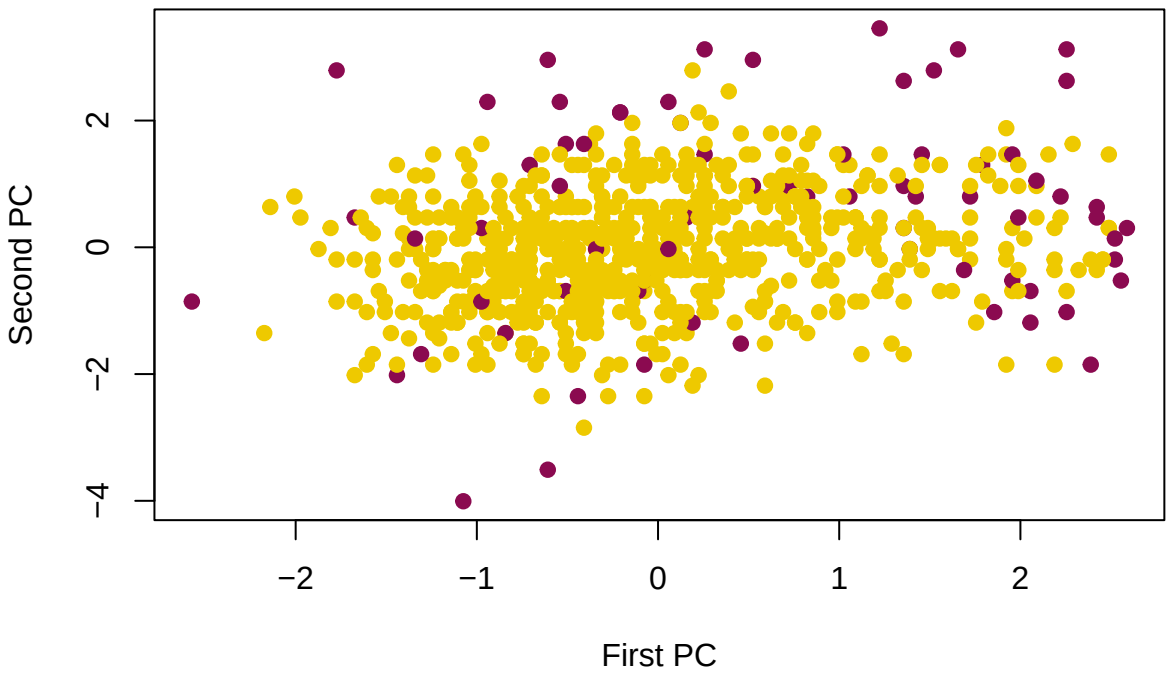
Ward linakge

DBSCAN

```
minPts <- 5
kNNdistplot(X2, k = minPts - 1)
abline(h = 1.8, col = c4)
```




```
db_fit <- dbscan(X2, eps = 1.8, minPts = 5)
colors_db <- c(c1, c2, c3)[db_fit$cluster + 1]
plot(X2[,1:2], pch = 19, col = colors_db, xlab = "First PC",
     ylab = "Second PC")
```




```
table(db_fit$cluster)
```

```
##
##   0   1
##  70 688
```

The clusters have very different sizes, and this is not a good indicator of a good split. The averages are:

	Cluster 1	Cluster 2
Glucose	140.0285714	119.3648256
BloodPressure	78.6142857	71.6889535
SkinThickness	31.3857143	28.4229651
LogPregnancies	1.1501942	1.3372052
LogInsulin	5.1157224	4.8019962
LogBMI	3.5375169	3.4818848
LogDPF	0.5235321	0.3441878
LogAge	3.6025736	3.4655986

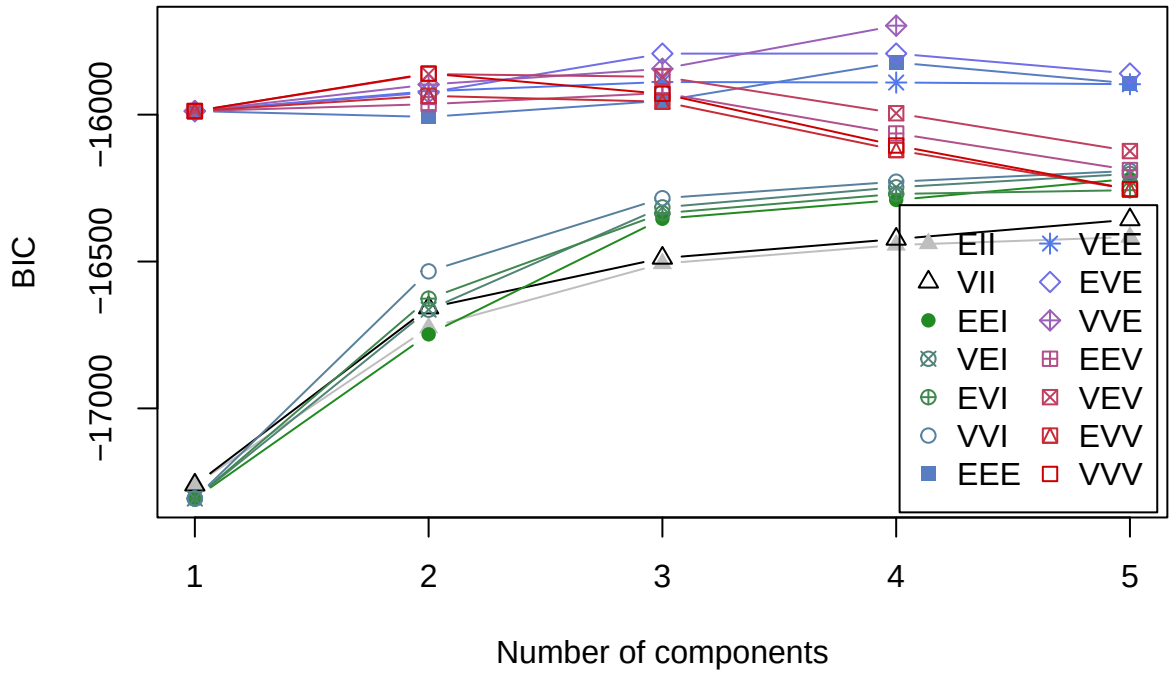
We can also see the confusion matrix with our target variable, although we must remember that this is not the objective of unsupervised learning.

```
kable(cluster_classification(db_fit$cluster, first = 0))
```

	Negative	Positive
Negative	457	231
Positive	37	33

Model-based clustering

```
BIC_X <- mclustBIC(X2, G = 1:5)
plot(BIC_X)
```




```
summary(BIC_X)
```

```
## Best BIC values:
##             VVE,4       EVE,4        EVE,3
## BIC      -15697.1 -15792.0269 -15792.25144
## BIC diff      0.0    -94.9267    -95.15127
```

The suggested number of clusters is 4. However, we can see that there is not much difference between choosing 2, 3, 4 and 5. The best models are EEE (ellipsoidal, equal volume, shape and orientation) with 4 clusters, and VEE (ellipsoidal, equal shape and orientation) with 4 and 5 clusters.

```
Mclust_X <- Mclust(X2, x = BIC_X, verbose = F)
summary(Mclust_X)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
```

```
## Mclust VVE (ellipsoidal, equal orientation) model with 4 components:
##
##  log-likelihood   n df      BIC       ICL
##       -7533.593 758 95 -15697.1 -15920.23
##
## Clustering table:
##   1   2   3   4
## 261 197 223  77
```

```
mb_means <- cluster_means(Mclust_X$classification, first = 1)
kable(mb_means)
```

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Glucose	132.8697318	104.2335025	122.9372197	120.7402597
BloodPressure	76.7854406	63.9695431	74.5874439	72.0649351
SkinThickness	30.2911877	25.4263959	29.4260090	29.5454545
LogPregnancies	1.4951560	0.9426618	1.9039043	0.0000000
LogInsulin	5.1186544	4.5389774	4.7464907	4.8475212
LogBMI	3.5145626	3.4200244	3.4869926	3.5651678
LogDPF	0.4930837	0.3452107	0.2352273	0.3154726
LogAge	3.7050818	3.2051317	3.5542757	3.1879379