Team Project

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

2023-01-17

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.

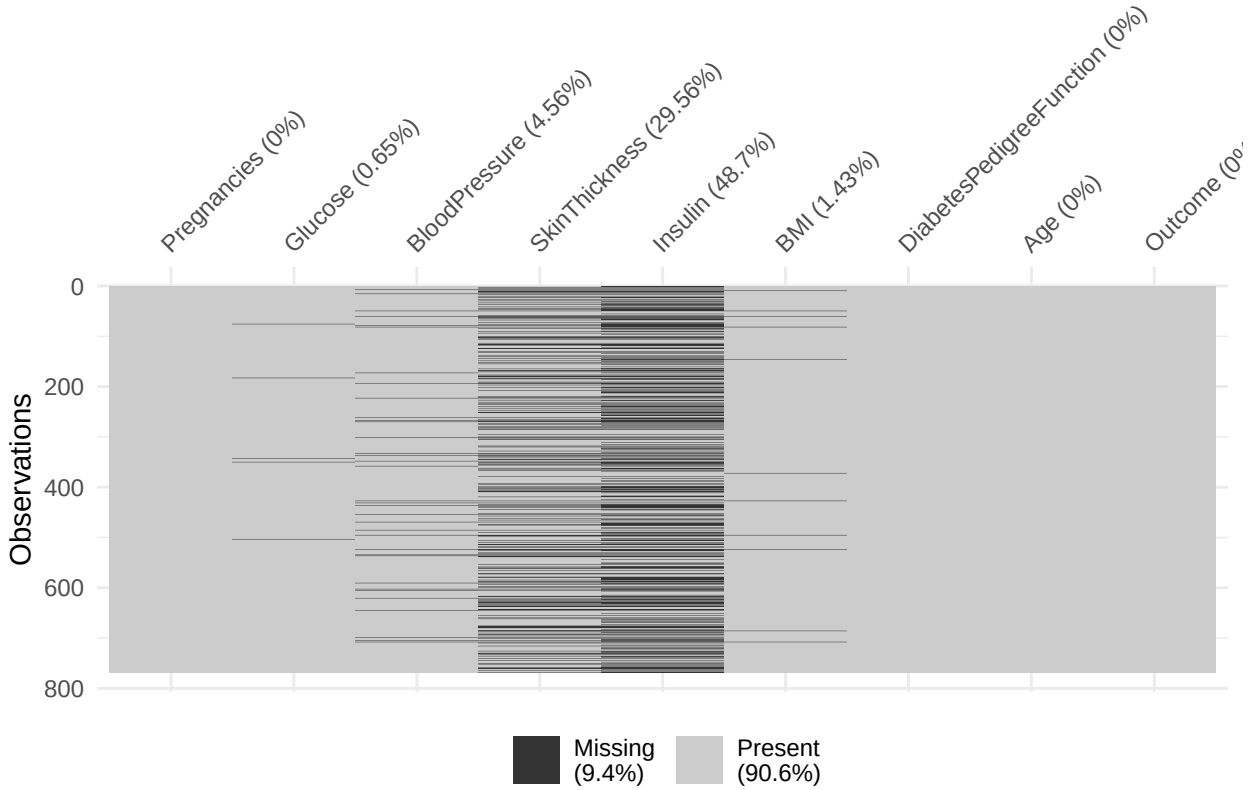
```
set_nas <- function(data, fields) {
    percentage <- list()
    for (field in fields) {
        data[[field]][data[[field]] == 0]   <- NA
        percentage[[field]] <- 100 * sum(is.na(data[[field]])) / nrow(data)
```

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

```
    }
    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.

```
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

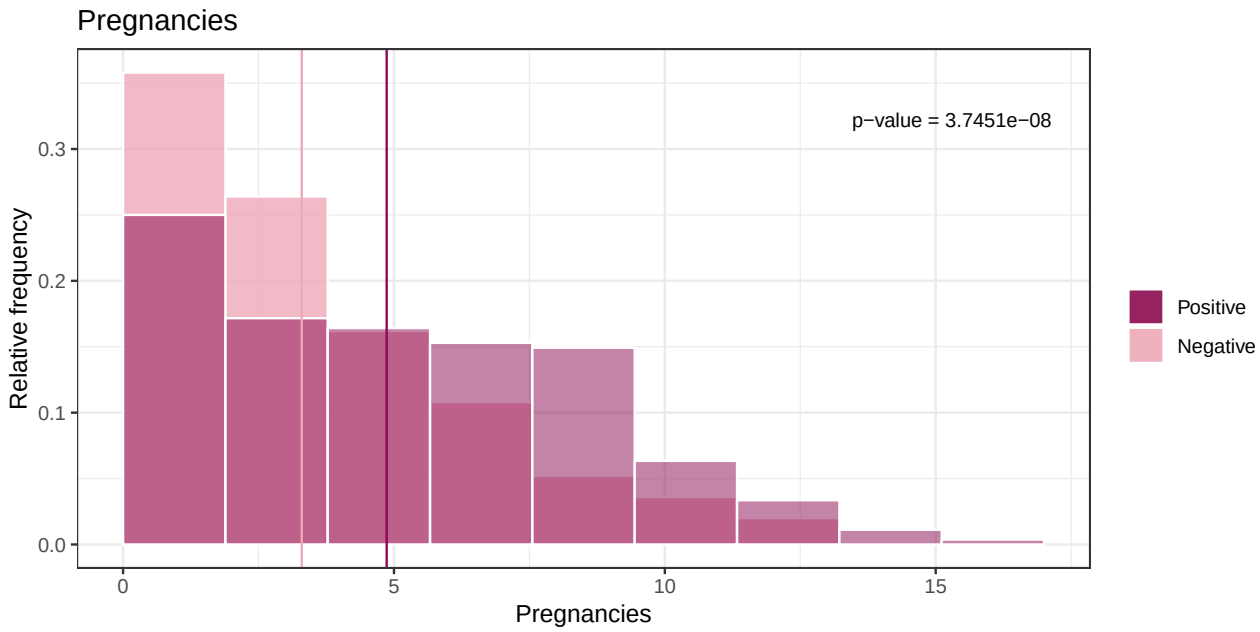
Before we proceed any further, we are going to describe our data. First, we look individually to each of the attributes, we see both visually and with with a two-sample Wilcox-test whether there is significant difference between the two groups (having or not diabetes), and if some transformation may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

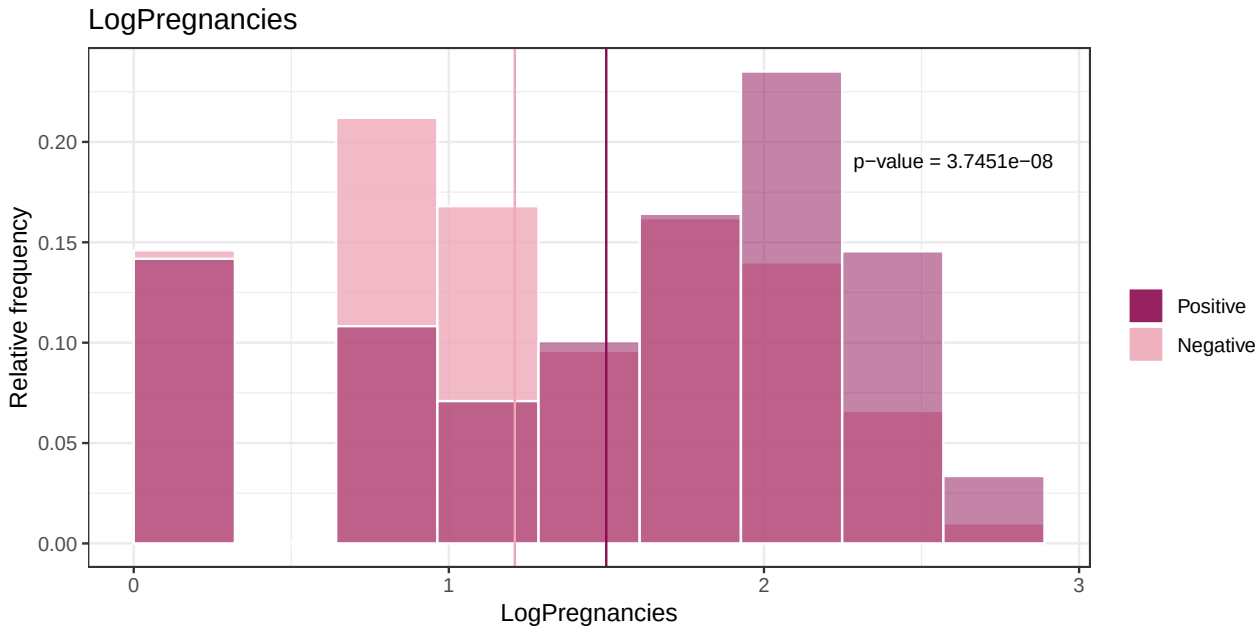
[2] Remember the domain of the logarithmic function is $(0, \infty)$.

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



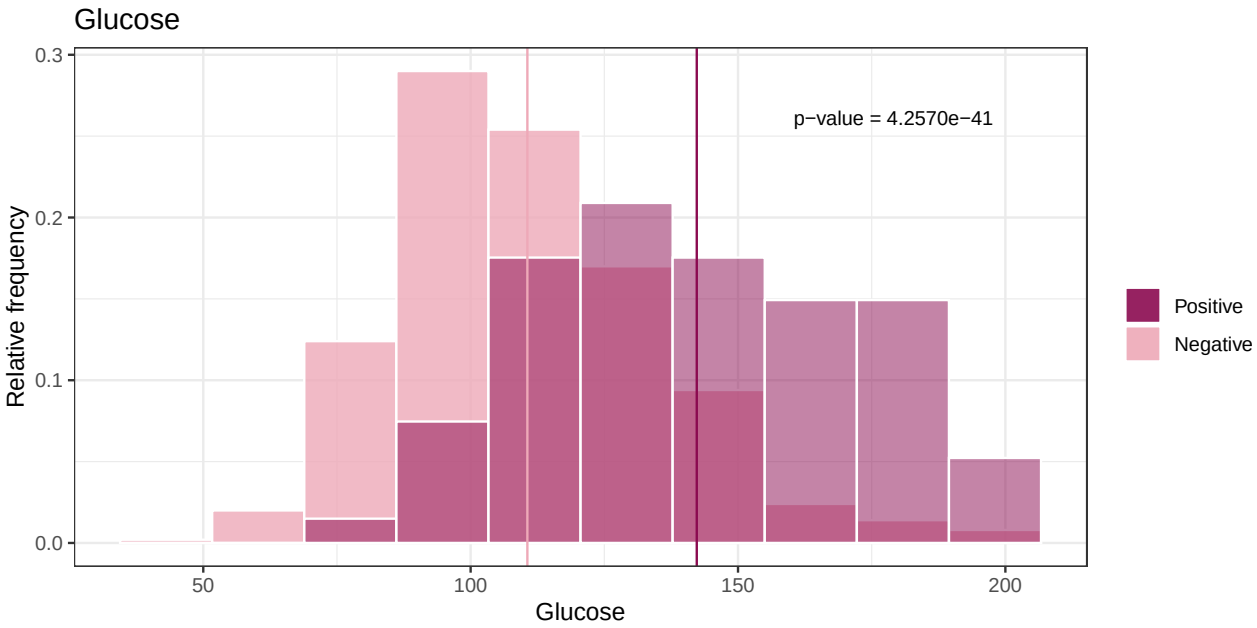
The change as we see does help in obtaining a more centered distribution with which we may better apply the posterior analysis.

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



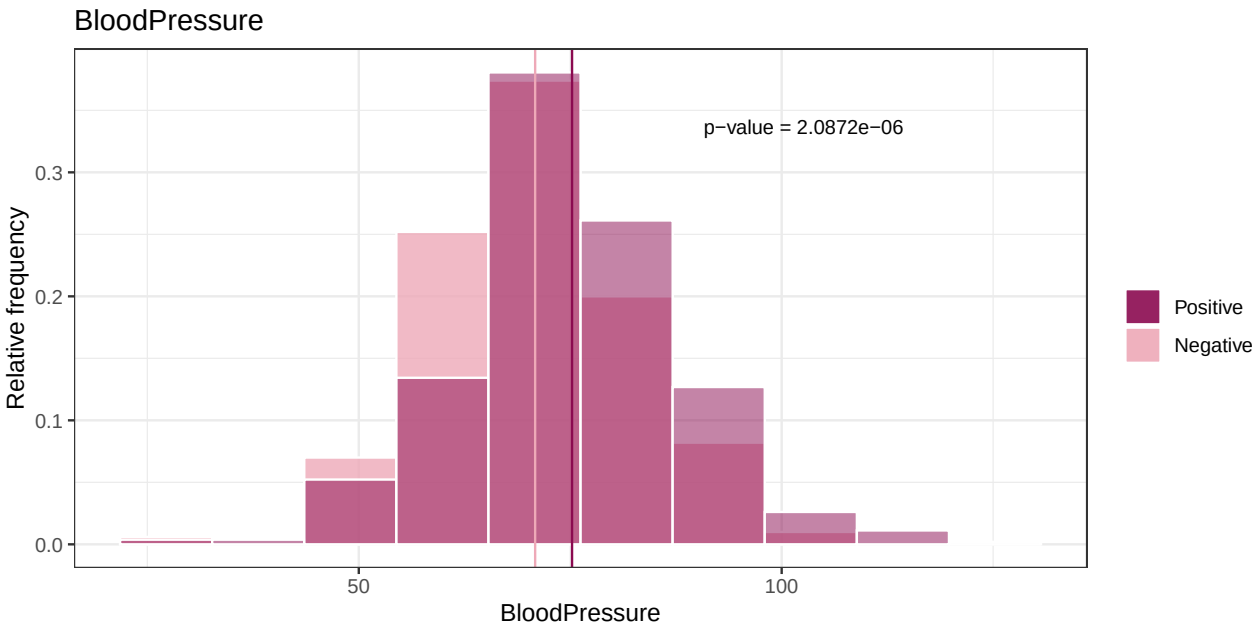
The next variable to visualize is the glucose. People with diabetes exhibit higher glucose values. The p-value is very small which indicates a highly significant difference between the two groups. We also observe that the distribution is already well-centered and resembles a normal distribution. Hence,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

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



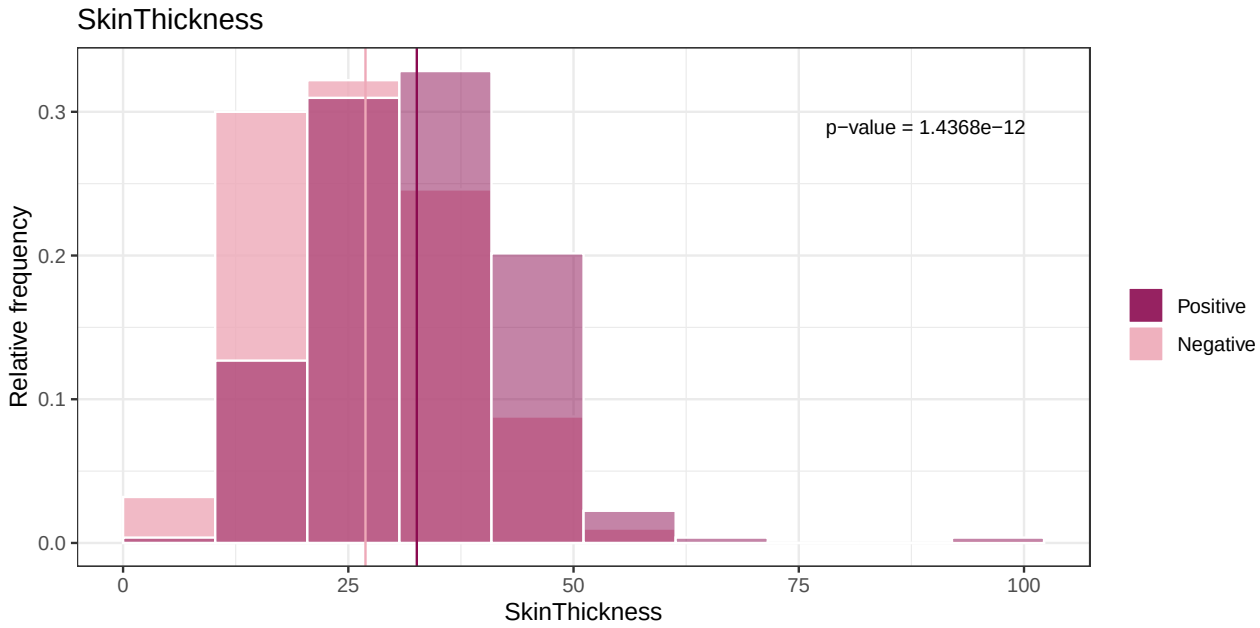
We move onwards to blood pressure. In this case, although the distributions appear as normal, there does not seem to exist a highly significant difference between the groups in contrast to what the p-value states, more so compared with the glucose variable. Nonetheless, there seem to be an slight indication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

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



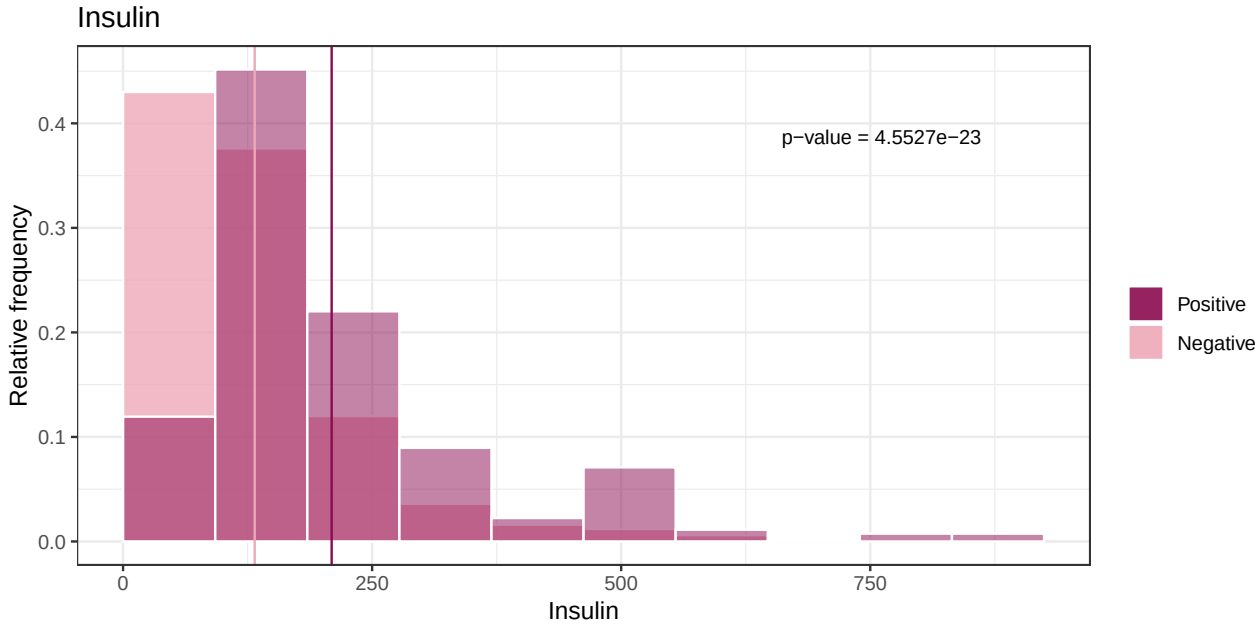
Now skin thickness is interesting, because the distributions are visually to those of the blood pressure but the presence of outliers is appreciable. We will later deal with those but for now let us mantain this variable as it is. Note that the population with diabetes appear to exhibit a thicker skin.

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



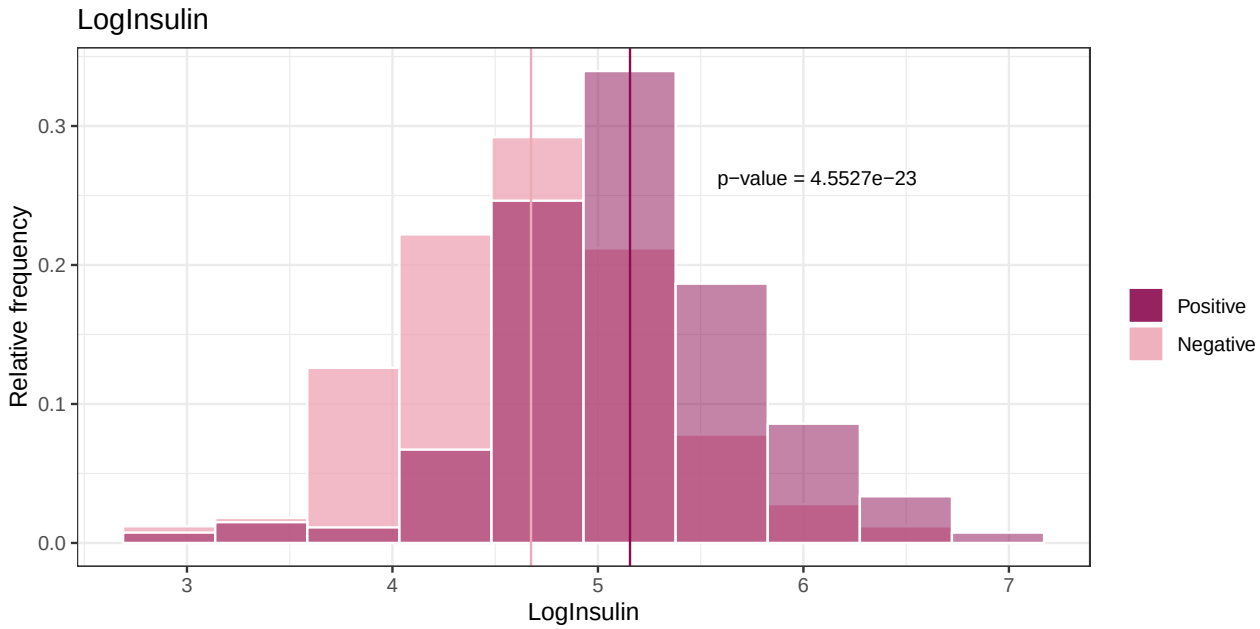
Insulin, as we may have expected from the name of the property itself, appears to be a relevant. The median of the distributions does indeed seem to differ, with the one for the diabetes population being slightly higher. It is also right-skewed, so we decide to log-transform it.

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



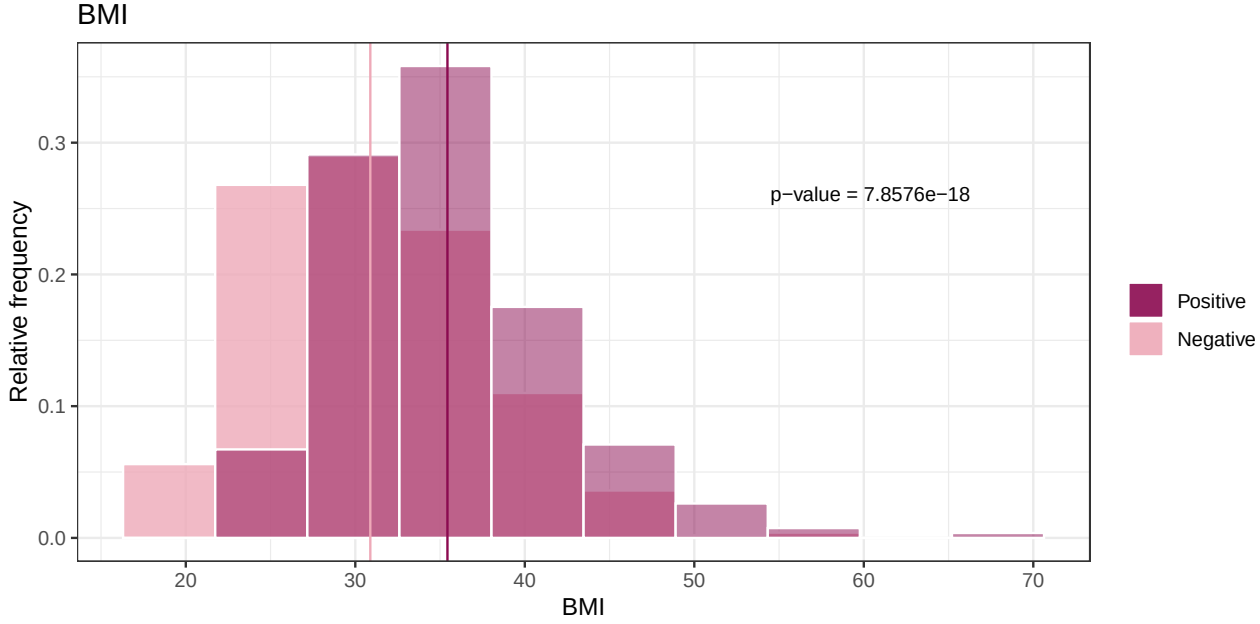
It does appear that the transformation impproves the the symmetrization of the data, although some left-skewness appears. We will later see if outlier detection get to target those values or not.

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



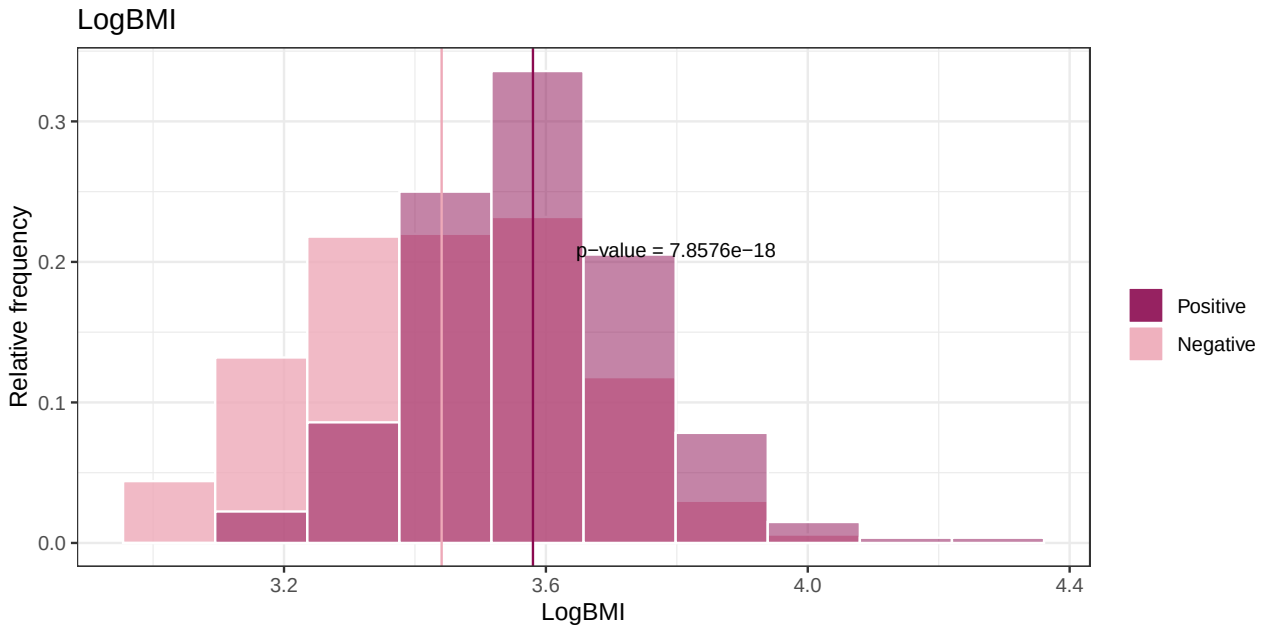
Body Mass Index (BMI) again exhibits this tendency of leaning towards a more right-skewed distribution. It does also follow the tendency of being slightly higher for people with diabetes. As such we log-transform to try and get a more normalized variable.

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



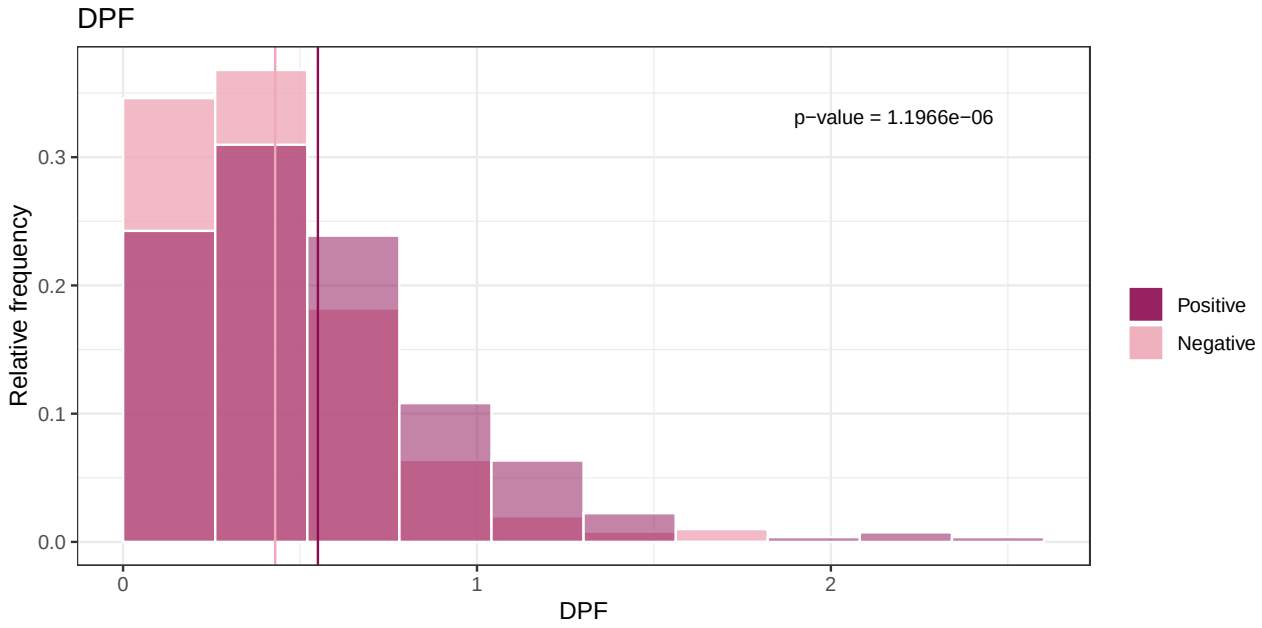
As we may visually judge, it is the case that the log transformation centers the data and provides a more normal-like distribution. There is some presence of seemingly outlying points to the right.

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



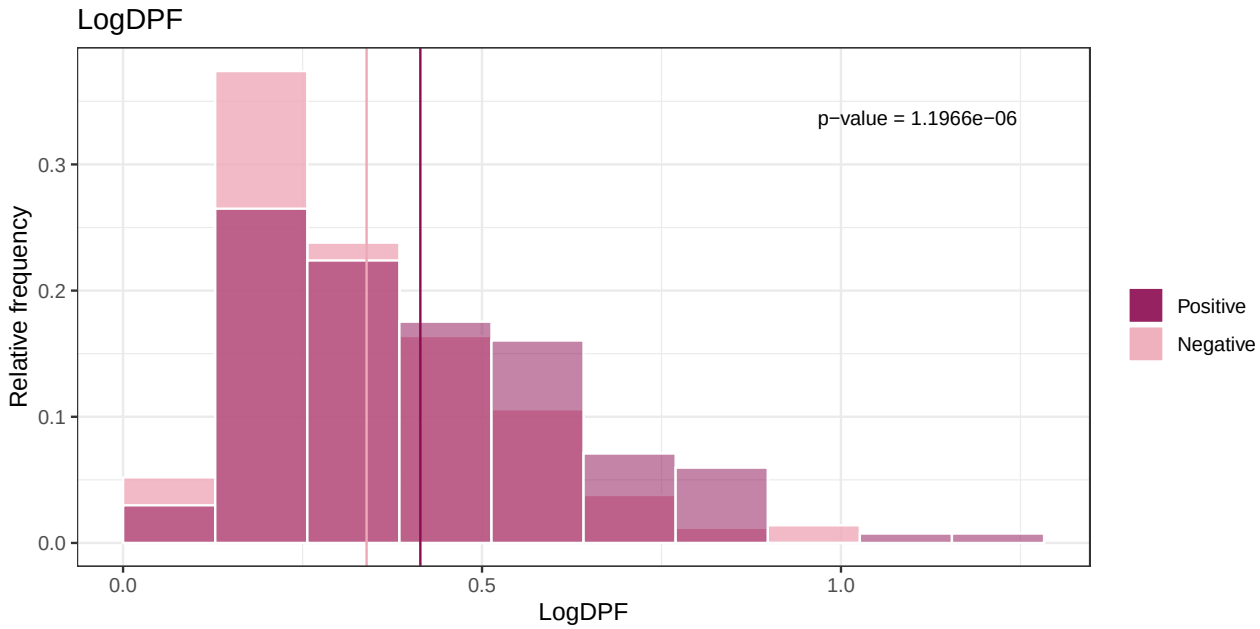
The Diabetes Pedigree Function (DPF) is the again a flagrant right-skewed. It again has higher values for the positive set. This is to be expected from the definition of this very function as a risk indication for diabetes. Let us try to log-transform it.

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



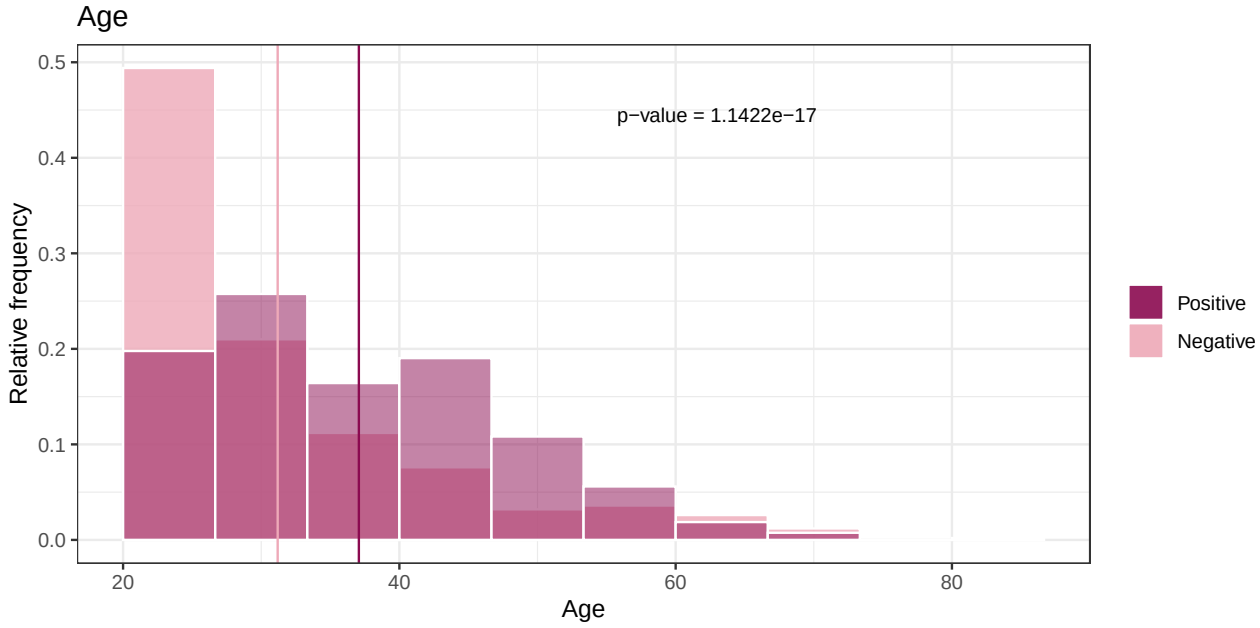
The improvement is highly noticeable. We retain thus this transformed variable.

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



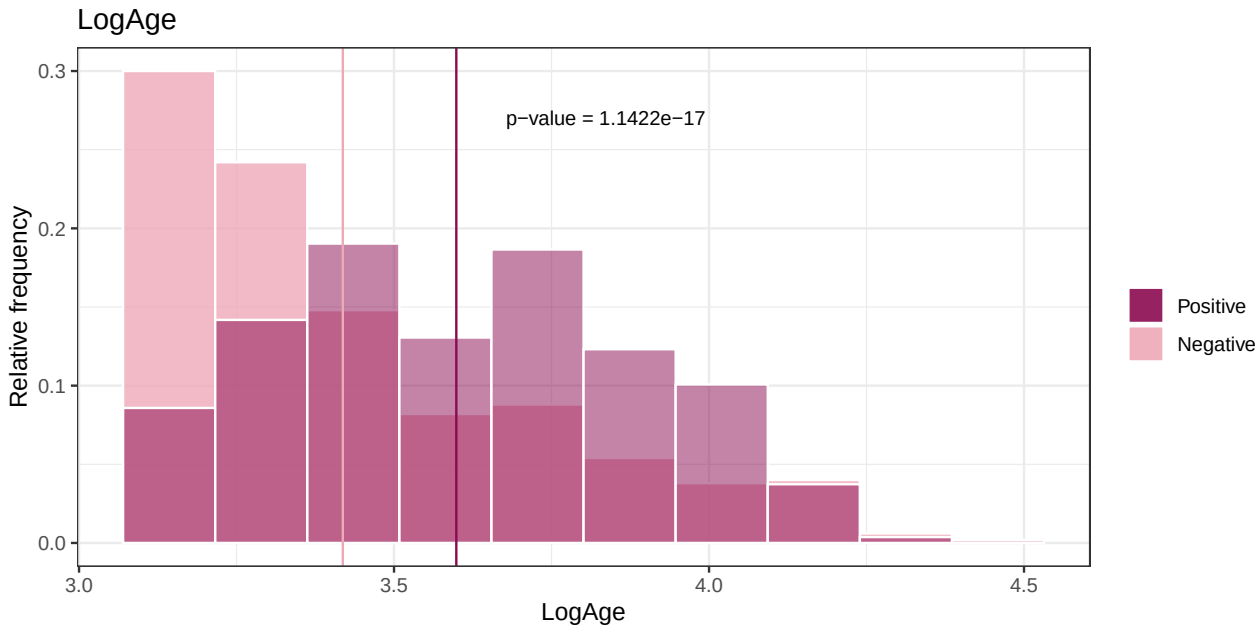
We may note that young people, as with other illnesses have less tendency to suffer diabetes than the elders. The age is expected to exhibit a right-skewed distribution, as is indeed the case. In an attempt to improve the symmetry, we once again use logarithms to transform the variable.

```
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.

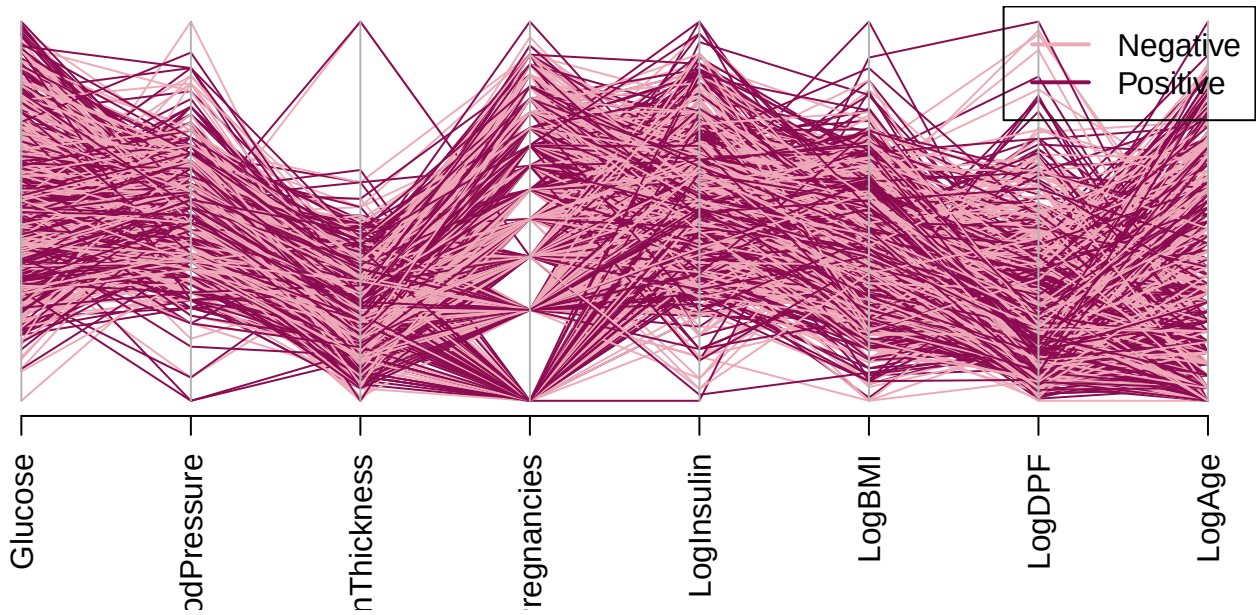
```
data$LogAge <- log(data$Age + 1)
histogram_by_groups(data, "LogAge")
```




```
# 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:

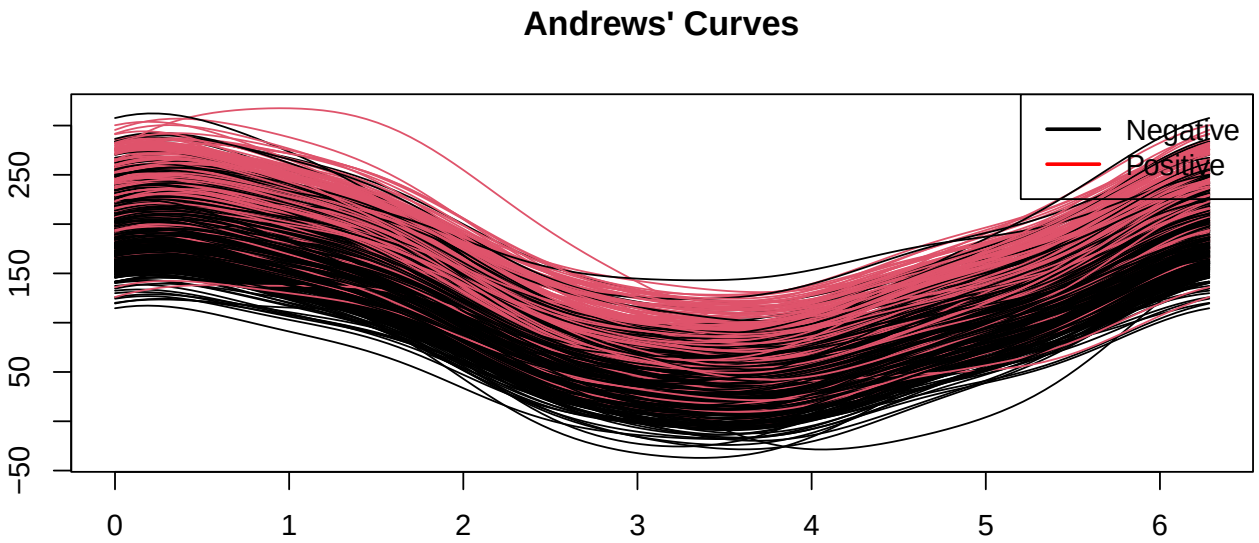
```
par(las = 2)
parcoord(df[xnames], col = c("pink2", "deeppink4"))
legend("topright", legend = c("Negative", "Positive"),
       col = c("pink2", "deeppink4"), lty = 1, lwd = 2)
```



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")

print("The mean vector is:")
```

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

```
print(mcd_est$center)
```

```
##         Glucose   BloodPressure   SkinThickness LogPregnancies      LogInsulin
##     120.0436620      71.8521127      28.5647887       1.3351757       4.8242845
##          LogBMI          LogDPF          LogAge
##       3.4842567       0.3525587       3.4642252
```

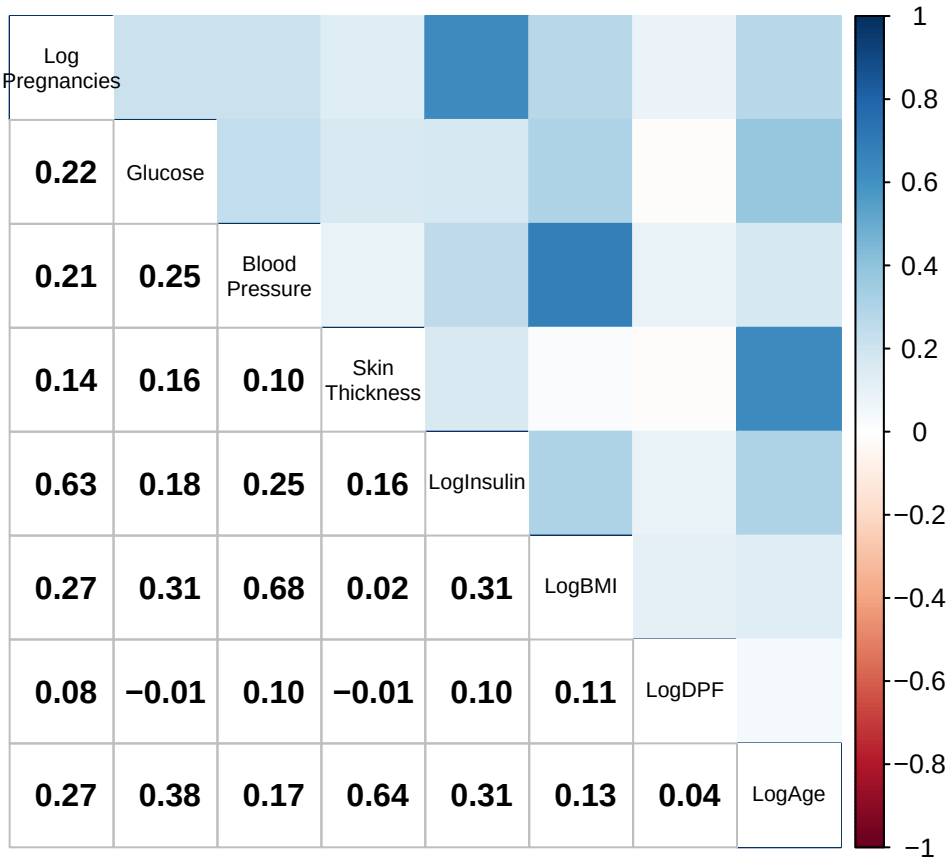
```
print("The covariance matrix is:")
```

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

```
print(mcd_est$cov)
```

```
##                     Glucose BloodPressure SkinThickness LogPregnancies
## Glucose         934.8734423   79.99669982    71.2537858    3.311934099
## BloodPressure    79.9966998  141.50719598    32.2981913    1.552817212
## SkinThickness    71.2537858   32.29819134   118.0164574    0.836657981
## LogPregnancies    3.3119341    1.55281721     0.8366580    0.628469749
## LogInsulin       13.2281360    1.42670355     1.8764597    0.086933966
## LogBMI            1.7183594    0.76099528     1.5384217    0.004023925
## LogDPF            0.4719628   -0.03400172     0.2028432   -0.001637966
## LogAge            2.6690444    1.43966056     0.6024776    0.161026776
##                   LogInsulin      LogBMI       LogDPF      LogAge
## Glucose         13.22813604 1.718359410  0.471962762 2.669044418
## BloodPressure    1.42670355 0.760995282 -0.034001720 1.439660561
## SkinThickness    1.87645973 1.538421715  0.202843181 0.602477562
## LogPregnancies   0.08693397 0.004023925 -0.001637966 0.161026776
## LogInsulin       0.46558177 0.043553473  0.012978939 0.066238970
## LogBMI           0.04355347 0.043123640  0.004399909 0.008720757
## LogDPF           0.01297894 0.004399909  0.036592522 0.002543337
## LogAge           0.06623897 0.008720757  0.002543337 0.101003429
```

```
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")

print("### Negative class ###")
```

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

```
print("The mean vector is:")
```

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

```
print(mcd_neg$center)
```

```
##        Glucose  BloodPressure  SkinThickness LogPregnancies     LogInsulin
##    107.9327354     70.1143498     26.6569507      1.2063315      4.6216510
##         LogBMI         LogDPF         LogAge
##      3.4379320      0.3224501      3.3760550
```

```
print("The covariance matrix is:")
```

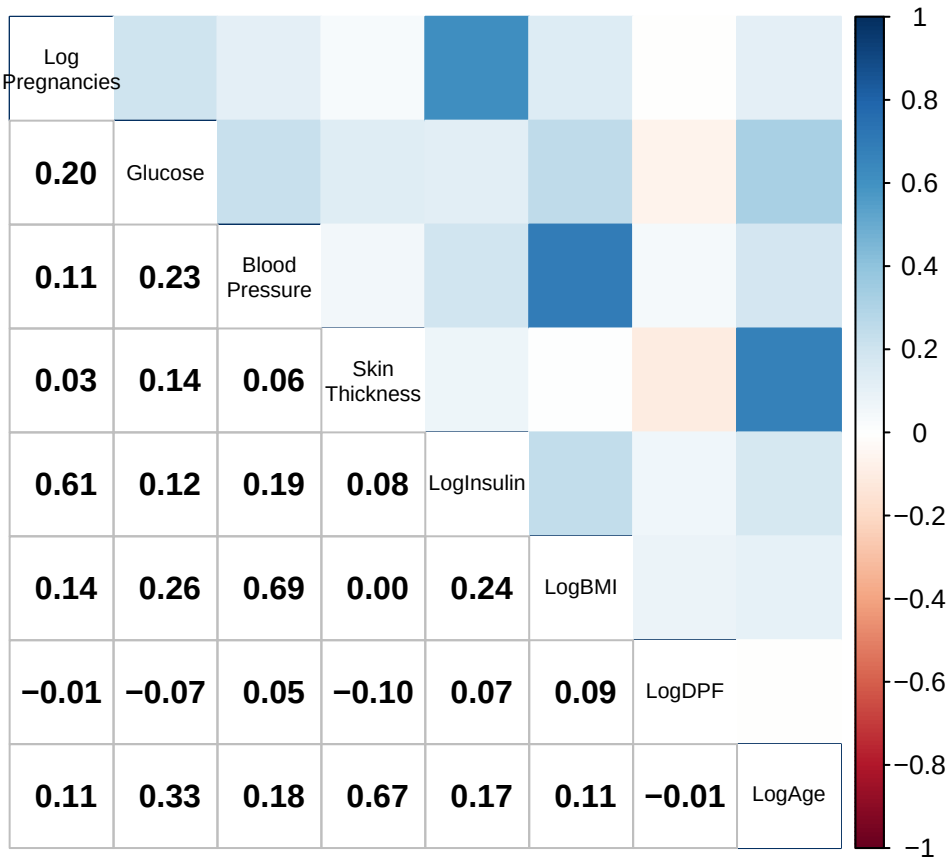
```
## [1] "The covariance matrix is:"
```

```
print(mcd_neg$cov)
```

```
##                    Glucose BloodPressure SkinThickness LogPregnancies
## Glucose        532.84017658    54.3903312   28.73488009   0.5390490522
## BloodPressure   54.39033123   133.2466584   28.39331536   1.2065688766
```

```
## SkinThickness   28.73488009    28.3933154  118.45217409   0.4785232866
## LogPregnancies   0.53904905     1.2065689    0.47852329   0.5840681703
## LogInsulin       9.43458093     0.9569082    1.39812050   0.0395727680
## LogBMI           0.69739228     0.6384802    1.61820214   0.0006428397
## LogDPF          -0.03421501    -0.1310348    0.08564691  -0.0130465983
## LogAge           0.75583081     1.0885927    0.56456771   0.1481947801
##                  LogInsulin       LogBMI        LogDPF        LogAge
## Glucose        9.434580926 0.6973922794 -0.0342150130  0.7558308089
## BloodPressure  0.956908188 0.6384802248 -0.1310348399  1.0885927160
## SkinThickness  1.398120496 1.6182021436  0.0856469058  0.5645677059
## LogPregnancies 0.039572768 0.0006428397 -0.0130465983  0.1481947801
## LogInsulin     0.443272315 0.0349783232  0.0075900057  0.0328213849
## LogBMI         0.034978323 0.0463017635  0.0032133966  0.0066293330
## LogDPF         0.007590006 0.0032133966  0.0289795853 -0.0002537536
## LogAge         0.032821385 0.0066293330 -0.0002537536  0.0829465371
```

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




```
print("### Positive class ###")
```

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

```
print("The mean vector is:")
```

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

```
print(mcd_pos$center)
```

```
##        Glucose  BloodPressure  SkinThickness LogPregnancies     LogInsulin
```

```
##    141.8207171     74.9482072     32.1553785      1.5290137      5.1640283
##         LogBMI         LogDPF         LogAge
##      3.5728380      0.4026572      3.5964355
```

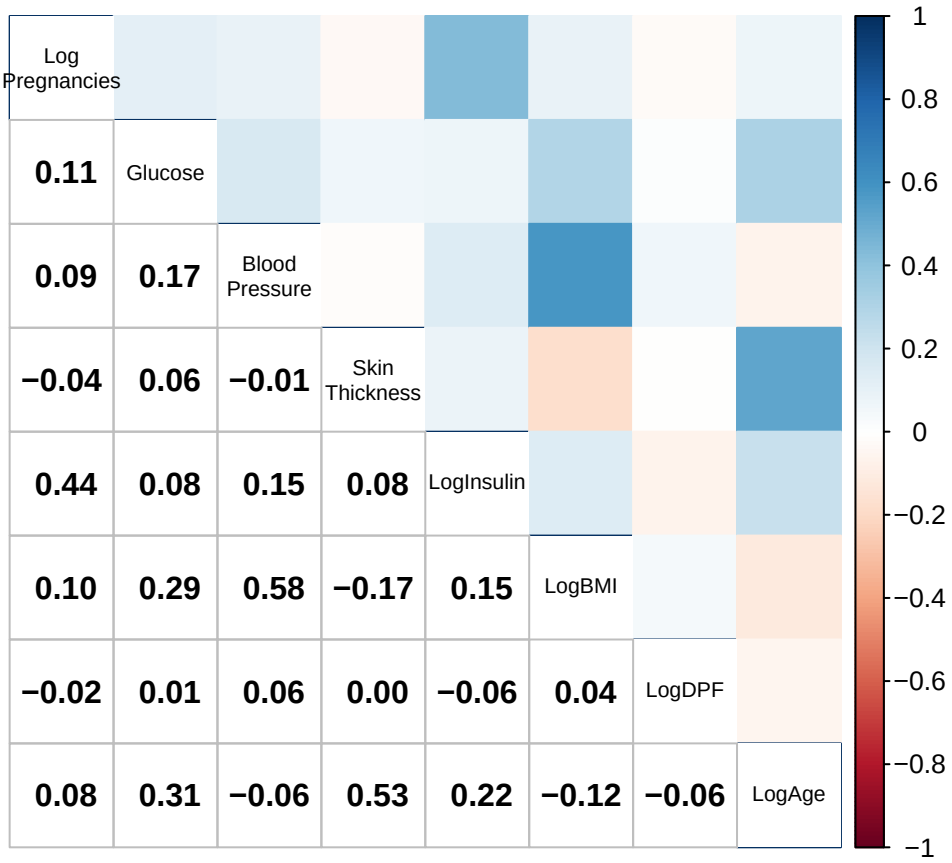
```
print("The covariance matrix is:")
```

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

```
print(mcd_pos$cov)
```

```
##                    Glucose BloodPressure SkinThickness LogPregnancies
## Glucose        948.1815958   42.82714414    29.0088906  -0.9854016352
## BloodPressure   42.8271441  150.48442005    21.0664634   0.6186188247
## SkinThickness   29.0088906   21.06646336   105.6064085  -0.1112158933
## LogPregnancies  -0.9854016    0.61861882    -0.1112159   0.6952181196
## LogInsulin       7.8616054    0.54611849     0.8852519   0.0398436405
## LogBMI           0.5153295    0.61332229     1.0174569  -0.0244192504
## LogDPF          -0.1428698    0.03000787     0.1302542  -0.0005170608
## LogAge           0.6940427    1.11209927    -0.1884792   0.1281449151
##                   LogInsulin       LogBMI        LogDPF       LogAge
## Glucose         7.861605374  0.515329507 -0.1428698188  0.694042734
## BloodPressure   0.546118490  0.613322290  0.0300078739  1.112099268
## SkinThickness   0.885251879  1.017456896  0.1302541815 -0.188479205
## LogPregnancies  0.039843641 -0.024419250 -0.0005170608  0.128144915
## LogInsulin      0.337194981  0.014573541 -0.0076308509  0.037477385
## LogBMI          0.014573541  0.028820026  0.0014233427 -0.005797771
## LogDPF         -0.007630851  0.001423343  0.0426701885 -0.003480916
## LogAge          0.037477385 -0.005797771 -0.0034809163  0.084957074
```

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



Le 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 $\mathcal{N}(\mu, \Sigma)$, 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

$$D_M(x, \mu)^2 > \chi^2_{p, 0.975^{1/n}} \ , \tag{1}$$

the 0.975^{th} quantile of the χ^2_p distribution. We then drop these outliers.

```
p   <- length(xnames)
n0  <- nrow(df0)
n1  <- nrow(df1)
df0_clean <- df0[mcd_neg$mah^2 < qchisq(0.975^(1 / n0), p), ]
df1_clean <- df1[mcd_pos$mah^2 < qchisq(0.975^(1 / n1), p), ]
df_clean  <- rbind(df0_clean, df1_clean)

print(paste("The negative set contained", n0 - nrow(df0_clean), "outliers."))
```

[3] Remember this is the logarithm.

```
## [1] "The negative set contained 319 outliers."
```

```
print(paste("The positive set contained", n1 - nrow(df1_clean), "outliers."))
```

```
## [1] "The positive set contained 176 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")
```