Repaso Examen: Predicción de la diabetes

Edmong Geraud - María Isabel Chuya - Nataly Quintanilla

NOTA: "LOS COMENTARIOS SE ENCUENTRAN EN ITEMS Y NEGRITA"

Intro

Este sería un ejemplo de examen El siguiente conjunto de datos, consiste en predecir a pacientes basandonos en datos clínicos, si puede padecer diabetes o no.

Antes de cualquier método de clasificación, regresión o lo que sea, necesitamos explorar los datos.

Esto supone exámenes estadísticos inferenciales univariantes, bivariantes y multivariantes.

- **Análisis univariado: Examina las características individuales de los datos clínicos.**
- **Análisis bivariado: Ayuda a comprender la relación entre las diferentes características y la variable objetivo.**
- **Análisis multivariante: Permite considerar simultáneamente varias características y evaluar su impacto en la predicción.**

Pima Indians Diabetes Database

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- **Este conjunto de datos proviene originalmente del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. El objetivo del conjunto de datos es predecir de manera diagnóstica si un paciente tiene o no diabetes, en función de ciertas medidas de diagnóstico incluidas en el conjunto de datos. Se impusieron varias restricciones a la selección de estas instancias de una base de datos más grande. En particular, todos los pacientes aquí son mujeres de al menos 21 años de herencia indígena pima.**

Cargamos librerias

- **Es recomendable cargar las librerías que se utilizarán para el analisis de clasificación o cualquier análisis al inicio, para que el programa corra correctamente.**

```
library(ggplot2)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(caret)
```

```
Loading required package: lattice
```

```
library(e1071)
library(ggstatsplot)
```

```
You can cite this package as:
     Patil, I. (2021). Visualizations with statistical details: The 'ggstatsplot' approach.
     Journal of Open Source Software, 6(61), 3167, doi:10.21105/joss.03167
```

```r
library(ggside)
```

```
Registered S3 method overwritten by 'ggside':
  method from
  +.gg   ggplot2
```

Cargamos los datos

- **Para cargar los datos se establece una variable y se lee los datos cargados en la misma carpeta "read.csv("./datos/diabetes.csv")"**
- **Se usa el comando "head()" para visualizar las primeras filas de un conjunto de datos o un objeto en forma de tabla.**

```r
datos <- read.csv("./datos/diabetes.csv")
head(datos)
```

```
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           6     148            72            35       0 33.6
2           1      85            66            29       0 26.6
3           8     183            64             0       0 23.3
4           1      89            66            23      94 28.1
5           0     137            40            35     168 43.1
6           5     116            74             0       0 25.6
  DiabetesPedigreeFunction Age Outcome
1                    0.627  50       1
2                    0.351  31       0
3                    0.672  32       1
4                    0.167  21       0
5                    2.288  33       1
6                    0.201  30       0
```

Si echamos una búsqueda rápida en google, observamos que el pedigree, es eso, la historia familiar de diabetes. Por lo tanto, aquí podríamso hacer varias cosas ! Entre ellas, regresar los datos a dicha función, o clasificar según esta variable, considerarla o no considerarla.

Para empezar vamos a considerarla para ver la clasificación del modelo knn y bayes.

Miramos las clases de los datos

- **El comando "str()" muestra un resumen conciso de la estructura de un objeto, incluyendo el nombre de las variables, sus tipos de datos y una muestra de los valores.**

```
str(datos)
```

```
'data.frame':   768 obs. of  9 variables:
 $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
 $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
 $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
 $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

La única variable que debemos de cambiar es `Outcome` a factor. Donde 1 es diebetes, y 0 es no diabetes

- **Para cambiar a factor una variable utilizamos el comando "as.factor()", es deceir "as.factor()" convierte la variable "Outcome" en un factor.**
- **Los valores de la variable, es decir los resultados son tratados como categorías o niveles en lugar de valores numericos, donde 1 indica diabetes, y 0 es no diabetes.**

```
datos$Outcome  <- as.factor(datos$Outcome)
```

Análisis estadístico preliminar

- **Para realizar el análisis estadístico preliminar veriificando el tamaño de datos usando el comando "dim(data)".**
- **La respuesta del comando es el número de filas y columnas en un conjunto de datos. Los datos tienen 768 filas y 9 columnas.**

```
dim(datos)
```

```
[1] 768   9
```

Tenemos 768 filas y 9 columnas. Analicemos primero dos a dos las variables una por una

Histogramas

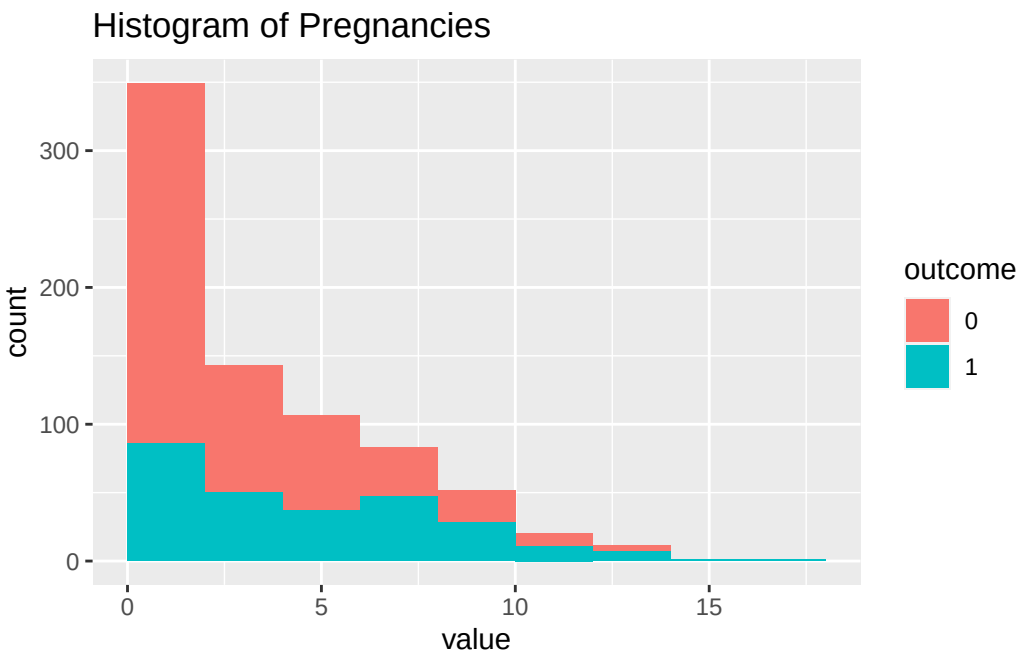
- **Se genera una lista con el comando "list" denominada la variable "l.plots" lo que permitirá graficar y guardar los histogramas.**
- **"n1" es una variable que engloba el número de columnas en el conjunto de datos menos uno, puesto que no se considera la variable "Outcome"**
- **Se logra iterar cada columna con el blucle "For", por esta razón se utiliza el comando "hist()", incluyendo el comando "plot" "FALSE" para no visualizar el histograma de inmediato.**
- **La variable "datos.tmp" contiene los valores de columna actuales incluyedo la varoable principal "Outcome"**

```
l.plots <- vector("list",length = ncol(datos)-1)
n1 <- ncol(datos) -1
for(j in 1:n1){

  h <-hist(datos[,j],plot = F)
  datos.tmp <- data.frame(value=datos[,j],outcome=datos$Outcome)
  p1 <- ggplot(datos.tmp,aes(value,fill=outcome))+geom_histogram(breaks=h$breaks) + ggtitl

  l.plots[[j]] <- p1
}
```

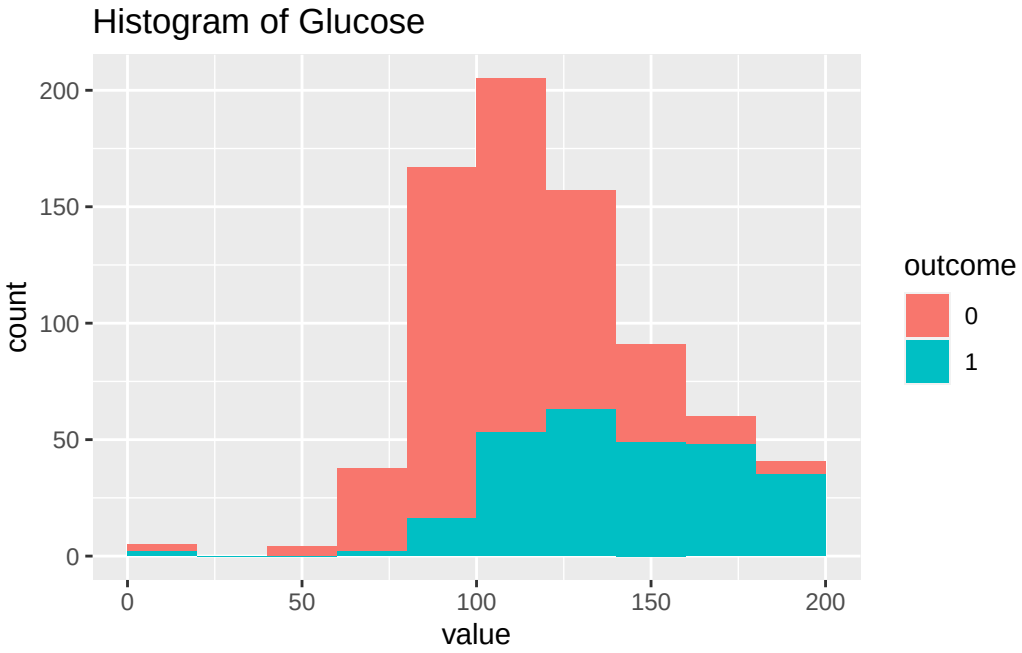
- **Se usa el comando "l.plots" para graficar los 8 histogramas generados para cada columna de "datos" excepto la columna principal "Outcome".**

```
l.plots
```

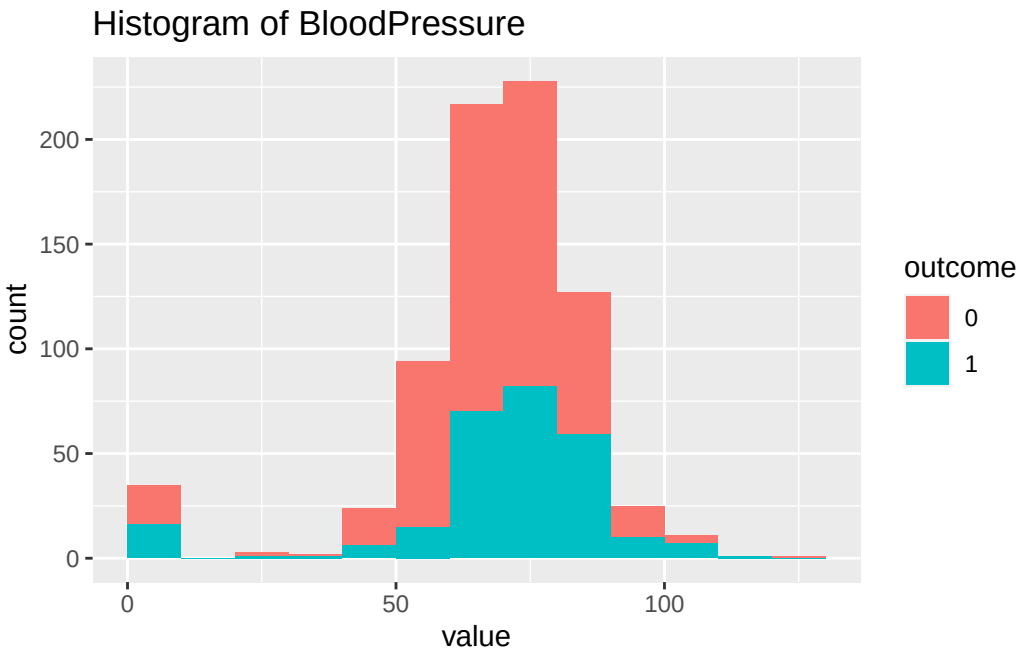
```
[[1]]
```


Histogram of Pregnancies
count
value
outcome
0
1
0
100
200
300
0
5
10
15

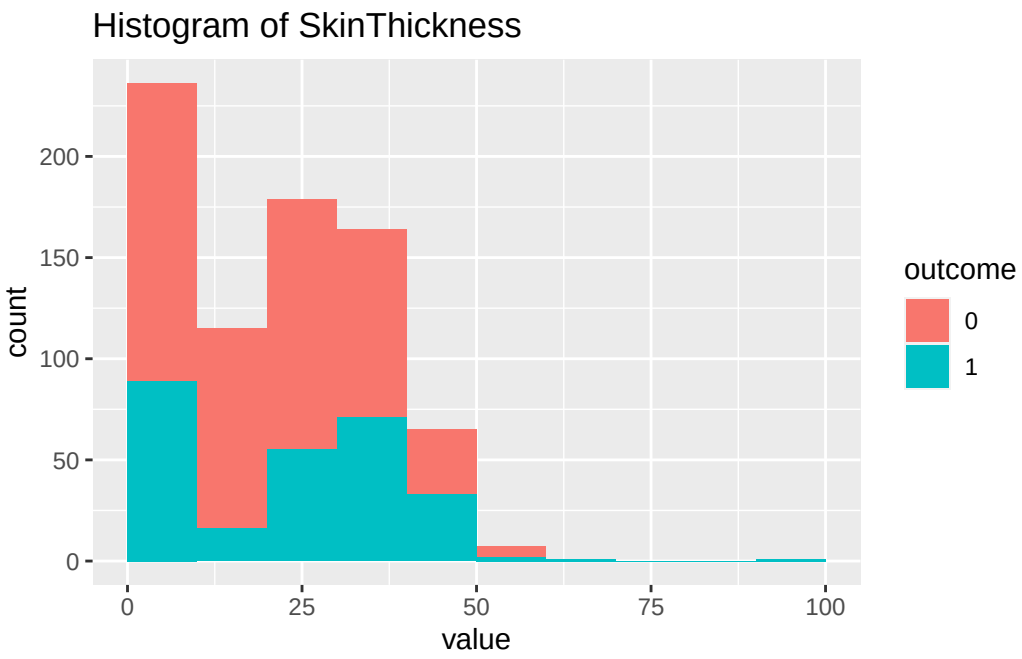
```
[[2]]
```


Histogram of Glucose
count
value
outcome
0
1
0
50
100
150
200
0
50
100
150
200

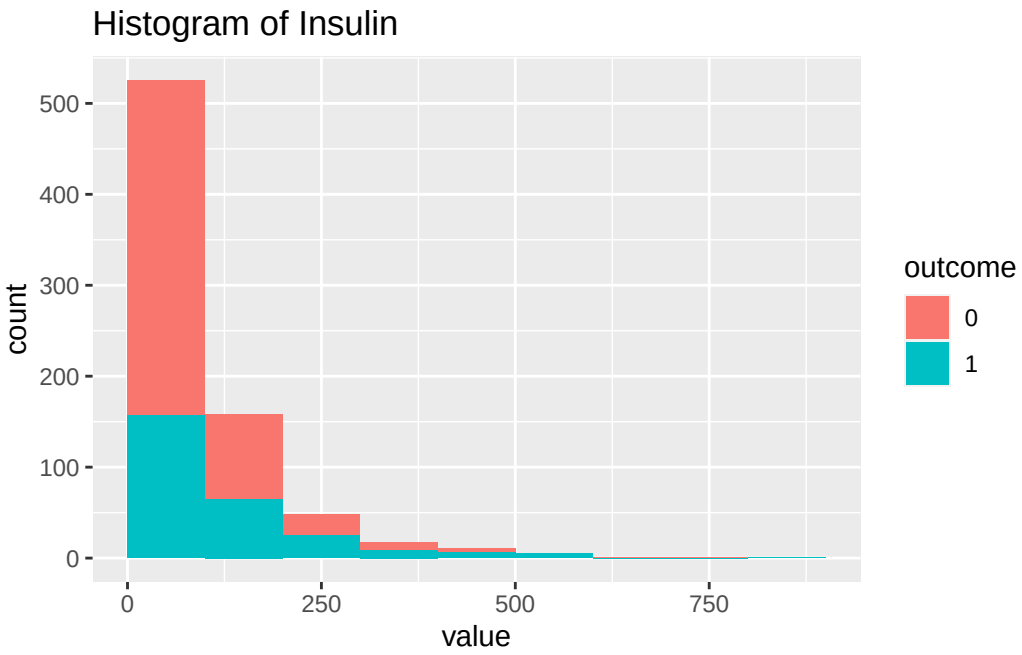
[[3]]


Histogram of BloodPressure
count
200
150
100
50
0
0
50
100
value
outcome
0
1

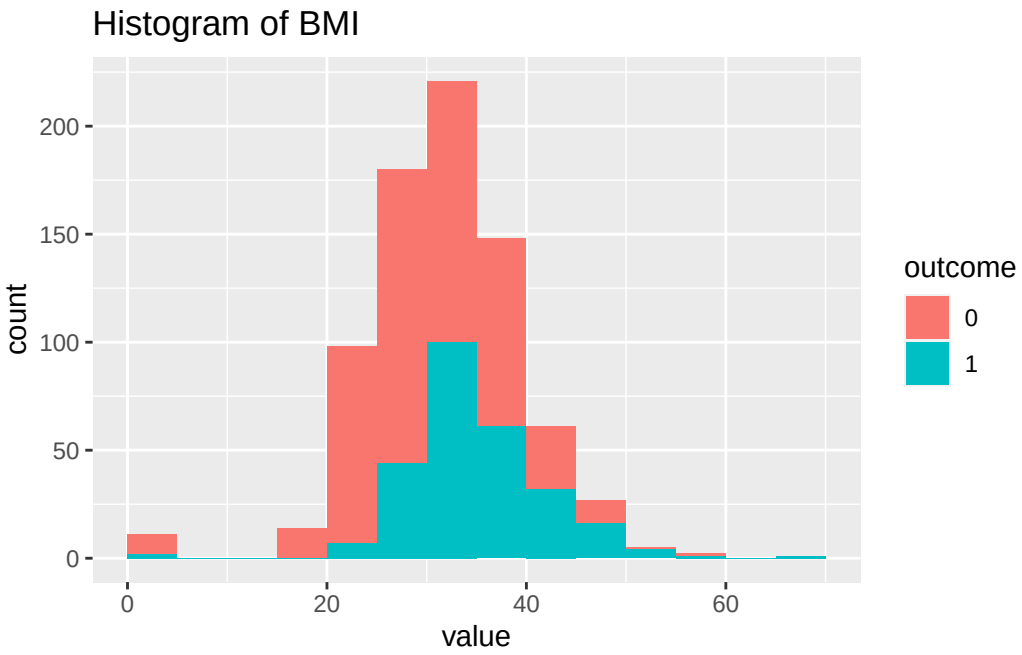
[[4]]


Histogram of SkinThickness
count
value
outcome
0
1
0
50
100
150
200
0
25
50
75
100

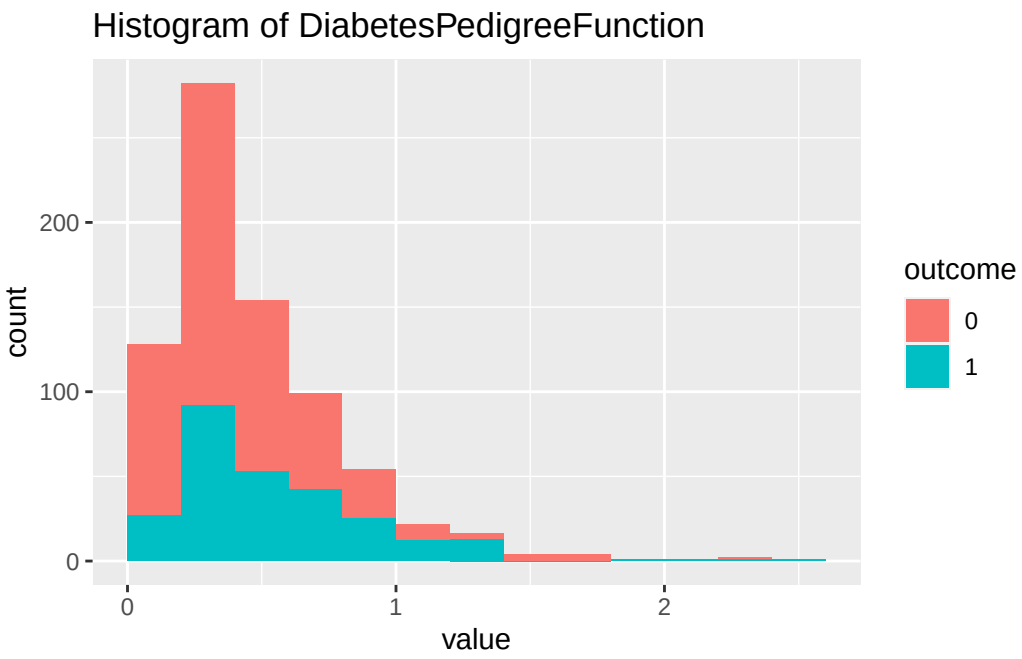
```
[[5]]
```


Histogram of Insulin
count
value
outcome
0
1
0
100
200
300
400
500
0
250
500
750

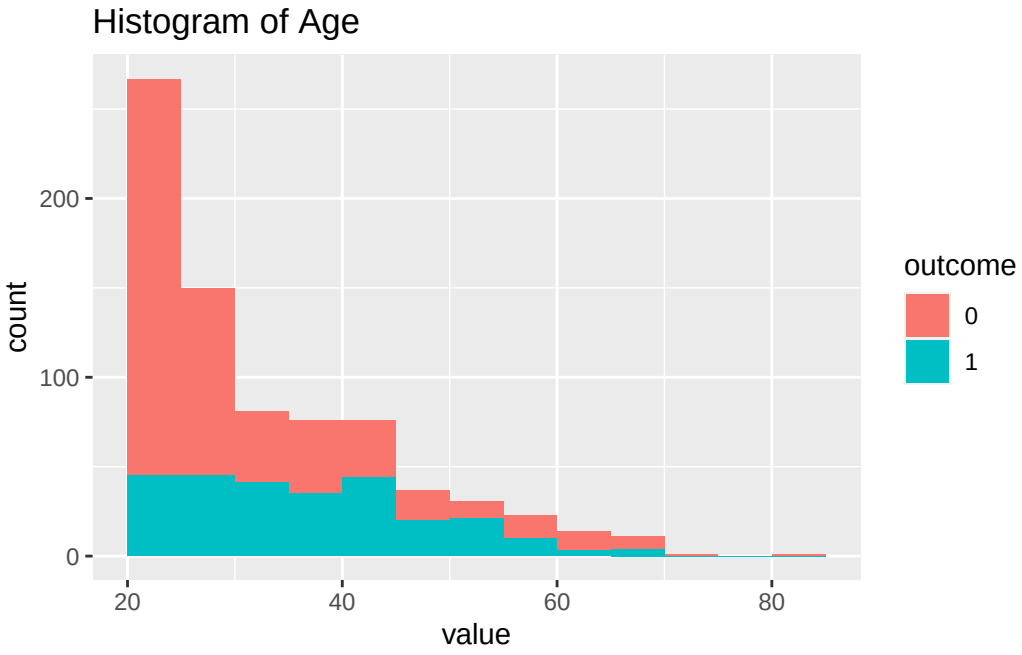
```
[[6]]
```


Histogram of BMI
count
200
150
100
50
0
0
20
40
60
value
outcome
0
1

```
[[7]]
```


Histogram of DiabetesPedigreeFunction
count
200
100
0
0
1
2
value
outcome
0
1

```
[[8]]
```


Histogram of Age
count
200
100
0
20
40
60
80
value
outcome
0
1

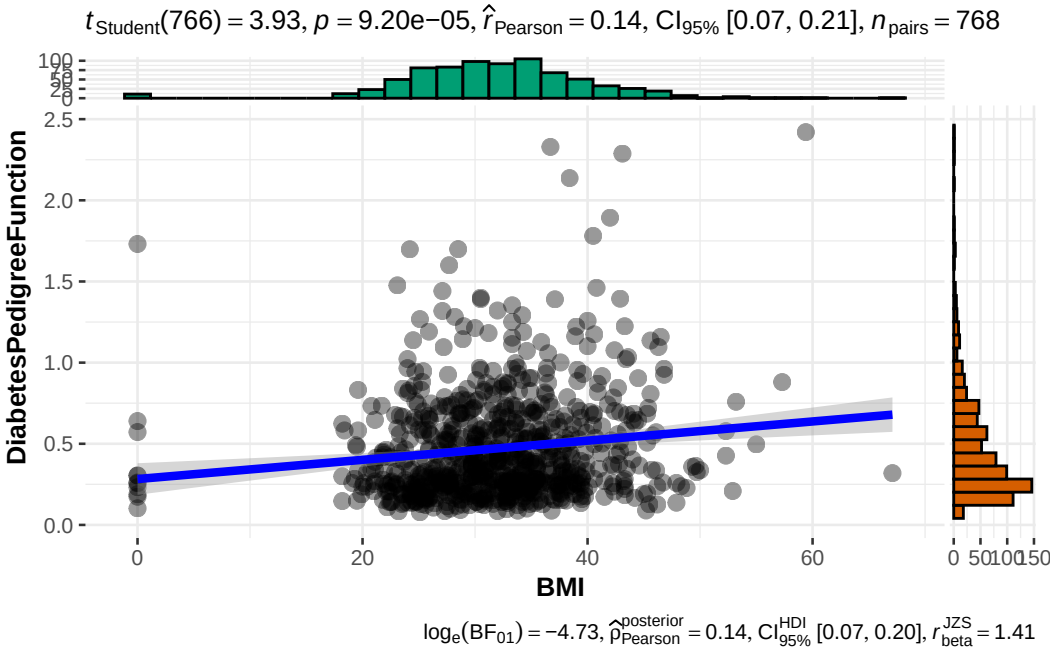
En lo particular la variable del pedigree se me hace importante, entonces vamos a realizar gráficos de dispersión

En realidad, una buena práctica es correlacionar todas contra todas…

- La función ggscatterstats(): Se usa para generar un gráfico de dispersión entre las variables "BMI" y "DiabetesPedigreeFunction".
- El gráfico muestra la relación entre ambas variables y también proporciona la correlación y los intervalos de confianza.***

```
ggscatterstats(datos,BMI,DiabetesPedigreeFunction)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

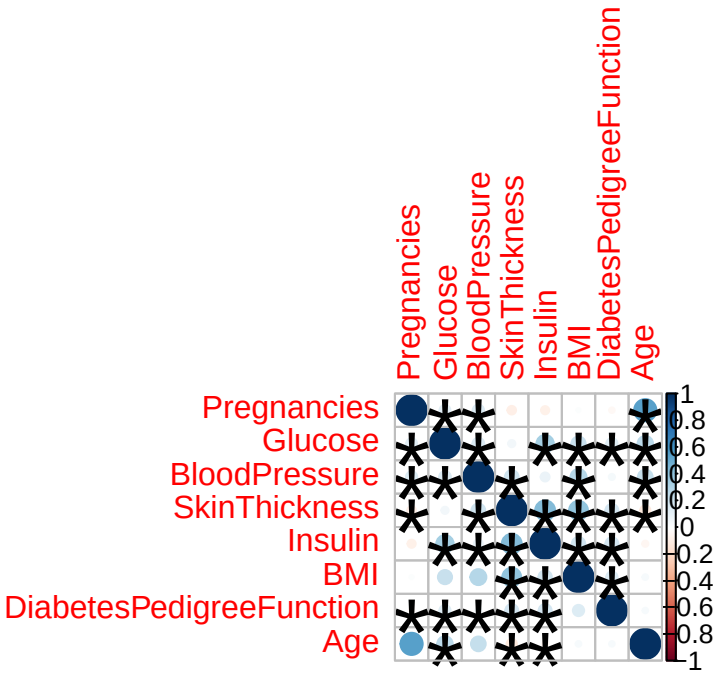


Sin embargo, esto puede ser un proceso tedioso… imaginad hacer 16 gráficas ! podemos condersarlo todo

- **Correlaciones entre las variables del conjunto de datos.**
- **La función "corr.test" correlación de las columnas del conjunto de datos desde la primera columna hasta la n1.**

- **El comando "upper" se usa para actualizar los valores de p en la parte superior de la matriz "p.values" con los valores ajustados obtenidos de "obj.cor$p.adj".**
- **El comando "upper.tri(p.values)" devuelve una matriz booleana con el mismo tamaño que "p.values"**
- **La función "corrplot" genera un gráfico de matriz de correlación.**
- **El comando "corr" la matriz de correlación obtenida a partir de "obj.cor$r".**
- **El comando "p.mat" la matriz de valores de p ajustados obtenida anteriormente.**
- **El comando "sig.level" el nivel de significancia utilizado para determinar qué correlaciones se consideran significativas.**
- **En "p.values" se guardan los valores de p obtenidos de la prueba de correlación realizada.**

```
obj.cor <- psych::corr.test(datos[,1:n1])
p.values <- obj.cor$p
p.values[upper.tri(p.values)] <- obj.cor$p.adj
p.values[lower.tri(p.values)] <- obj.cor$p.adj
diag(p.values) <- 1
corrplot::corrplot(corr = obj.cor$r,p.mat = p.values,sig.level = 0.05,insig = "label_sig")
```



Ahora podemos proceder a hacer algo similar, con una serie de comparaciones dos a dos sobre las medias o medianas, sobre cada variable y la variable de interés.

Primero debemos aplicar una regresión linear con variable dependiente cada variable numérica y por la categórica. Es decir un t.test pero con el fin de ver los residuos, para ver la normalidad de éstos

- **Análisis de normalidad de los residuos utilizando la prueba de "Shapiro-Wilk"**
- **Se aplica una regresión lineal donde "x" es cada columna de datos y "datos$Outcome" es la variable númerica en relación de la respuesta, utilizando "lm(x~datos$Outcome)".**
- **Se calculan y extraen los residuos de cada regresión usando el modelo: "summary(lm(x~datos$Outcome))$residuals)"**
- **La función 'apply()" se utiliza dos veces para aplicar la prueba de Shapiro-Wilk a los residuos de cada regresión "(shapiro.test)"**
- **Se utiliza "apply(datos[,1:n1], 2,...)" para aplicar la función a cada columna del conjunto de datos "datos[,1:n1]"**
- **El resultado se almacena en "p.norm", esto devuelve una matriz donde cada columna contiene los residuos del modelo ajustado para cada columna de datos, contiene los valores de p de las pruebas de normalidad realizadas con los residuos de cada modelo.**

```r
p.norm <- apply(apply(datos[,1:n1],
            2,
            function(x) summary(lm(x~datos$Outcome))$residuals),
      2,
      shapiro.test)

p.norm
```

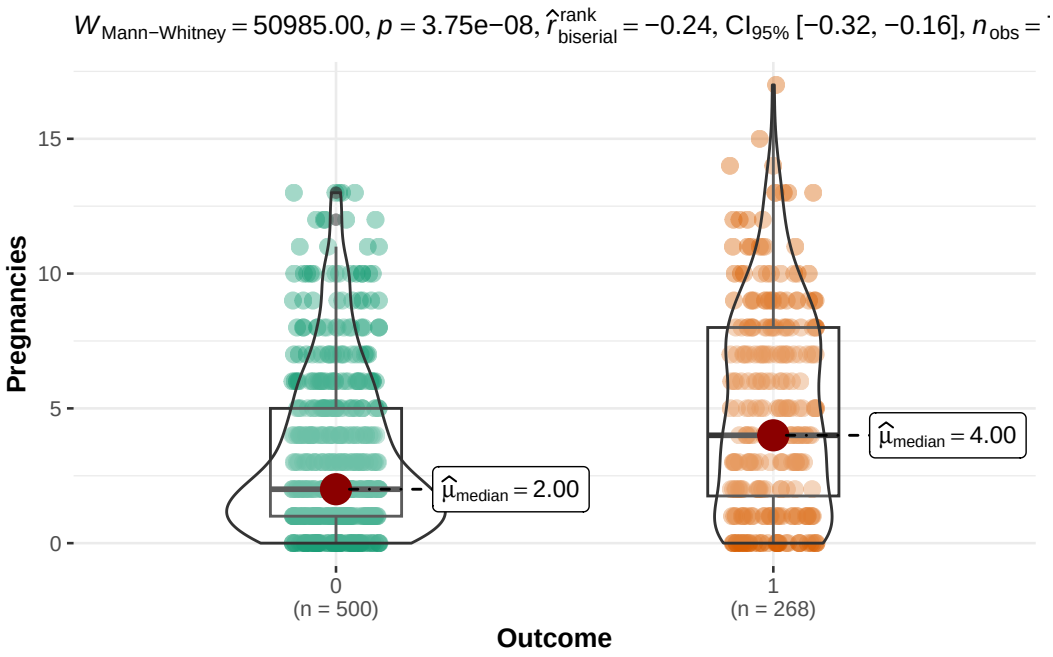
```
$Pregnancies

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.9389, p-value < 2.2e-16


$Glucose

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.97511, p-value = 3.726e-10


$BloodPressure

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.81468, p-value < 2.2e-16


$SkinThickness

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.92004, p-value < 2.2e-16


$Insulin
```

```
    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.77776, p-value < 2.2e-16


$BMI

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.94359, p-value < 2.2e-16


$DiabetesPedigreeFunction

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.84939, p-value < 2.2e-16


$Age

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.88114, p-value < 2.2e-16
```

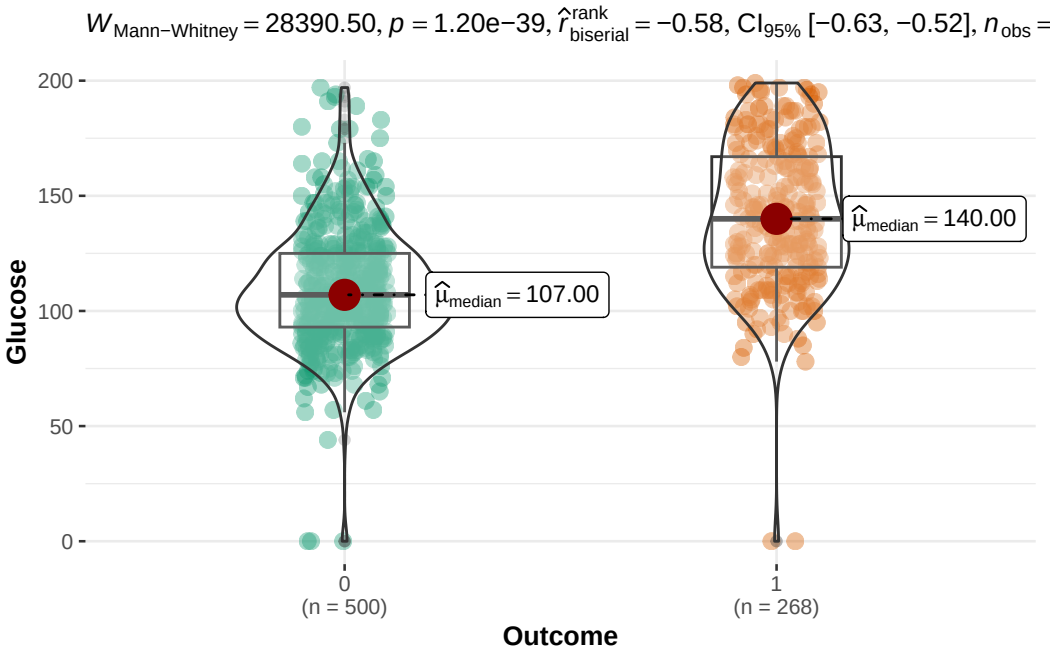
Todas las variables son no normales, tal como vemos en los histogramas.

- **El comando "ggbetweenstats()" realiza una comparación entre las variables "Pregnancies" y "Outcome" (pruebas no paramétricas).**
- **El gráfico muestra la distribución de la variable "Pregnancies" para cada categoría de la variable "Outcome" (diabetes o no diabetes).**
- **El tipo de prueba no paramétrica se especifica mediante el comando "type ="nonparametric"".**

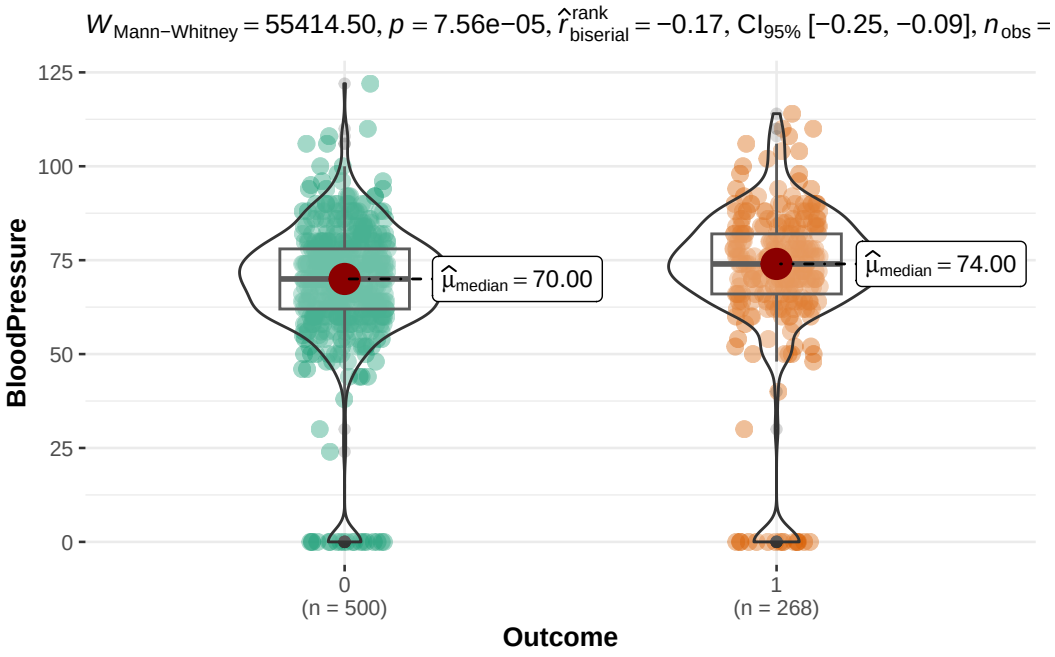
```
ggbetweenstats(datos,Outcome,Pregnancies,type = "nonparametric")
```



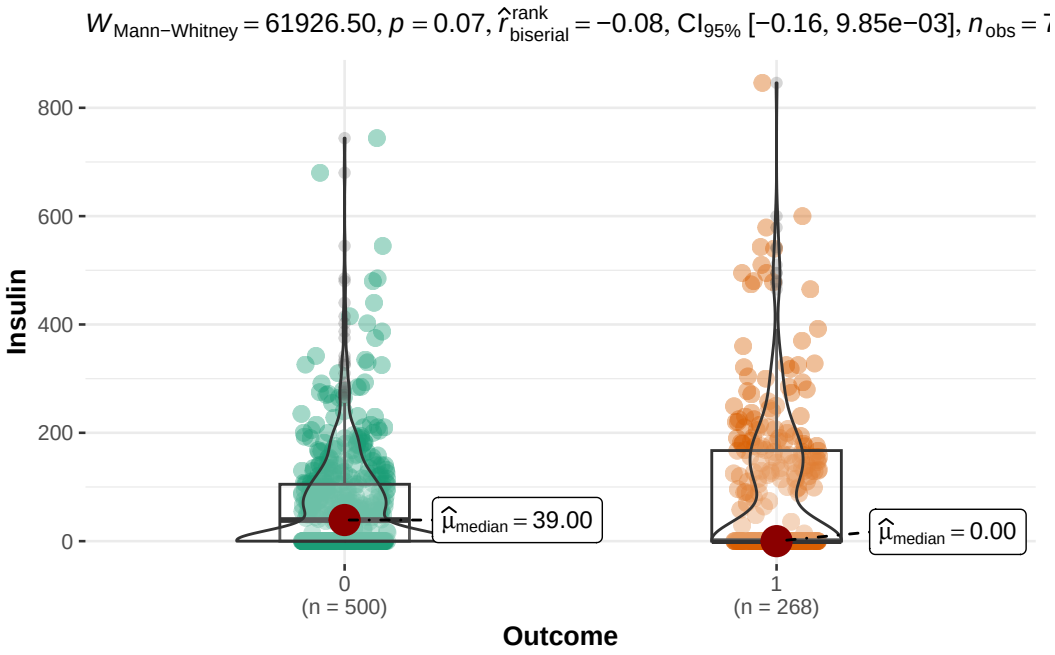

```
ggbetweenstats(datos,Outcome,Glucose,type = "nonparametric")
```



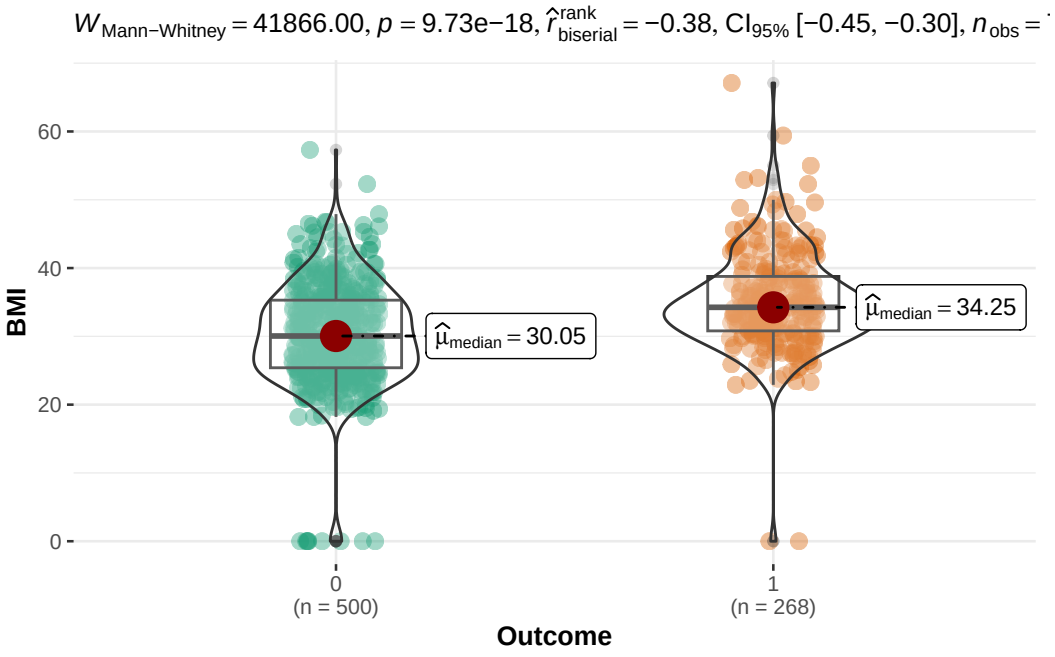
```
ggbetweenstats(datos,Outcome,BloodPressure,type = "nonparametric")
```



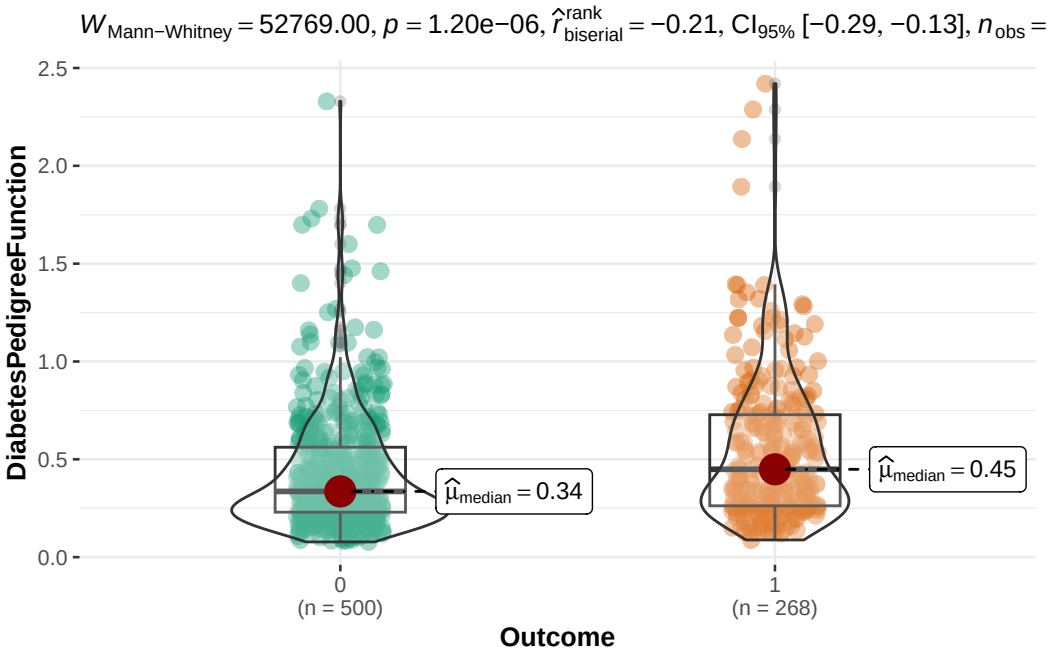

```
ggbetweenstats(datos,Outcome,Insulin,type = "nonparametric")
```



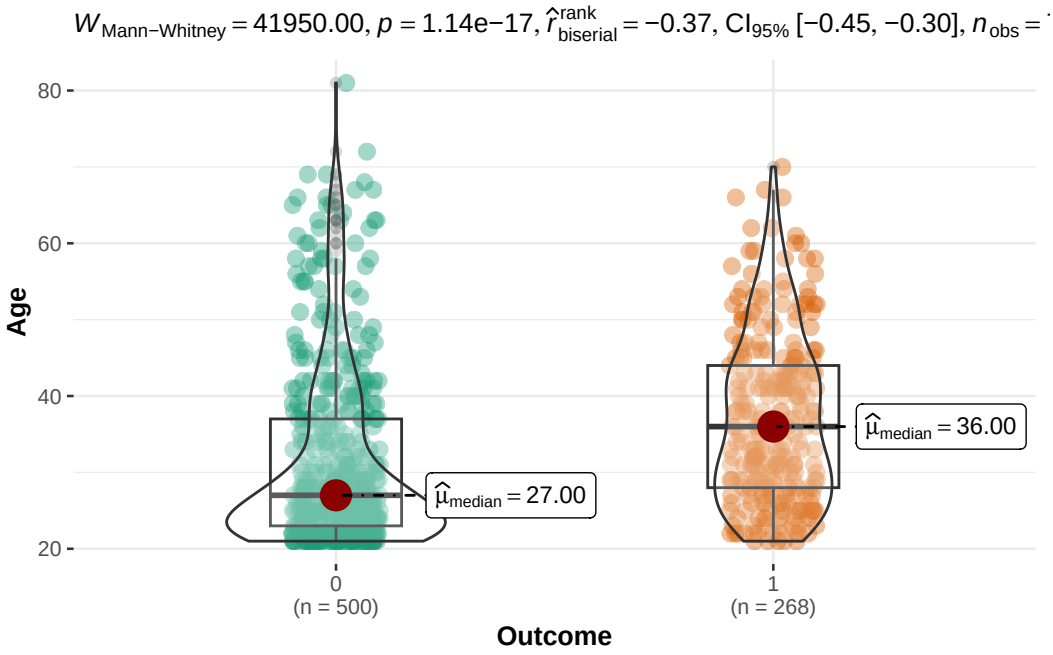

```
ggbetweenstats(datos,Outcome,BMI,type = "nonparametric")
```



```
ggbetweenstats(datos,Outcome,DiabetesPedigreeFunction,type = "nonparametric")
```




```
ggbetweenstats(datos,Outcome,Age,type = "nonparametric")
```



PCA

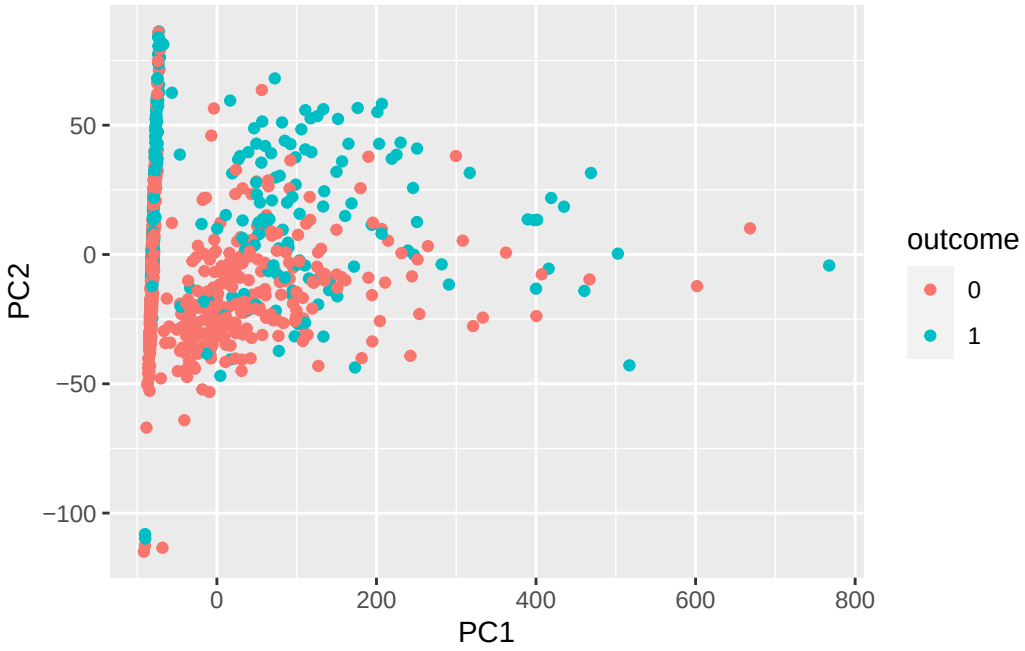
- **La función "prcomp" se usa para calcular las componentes principales.**
- **El comando "scale. = F" indica que no se deben escalar los datos por la variabilidad.**
- **Después de realizar PCA, tome los valores de los componentes principales "(pcx$x)" y combínelos con la variable "Resultado" del conjunto de datos "(datos$Resultado)" usando la función "bind_cols()" para crear un nuevo conjunto de datos llamado plotca.**
- **Se utiliza "ggplot()" para generar un gráfico de dispersión de (PC1 y PC2) .**

```
summary(datos)
```

```
  Pregnancies        Glucose      BloodPressure    SkinThickness
Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
```

```
   Insulin           BMI        DiabetesPedigreeFunction      Age
Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
Outcome
0:500
1:268
```

```r
pcx <- prcomp(datos[,1:n1],scale. = F) ## escalamos por la variablidad de los datos

plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()
```



Ahora vamos a ver si haciendo unas transformaciones esto cambia. Pero antes debemos de ver las variables sospechosas...

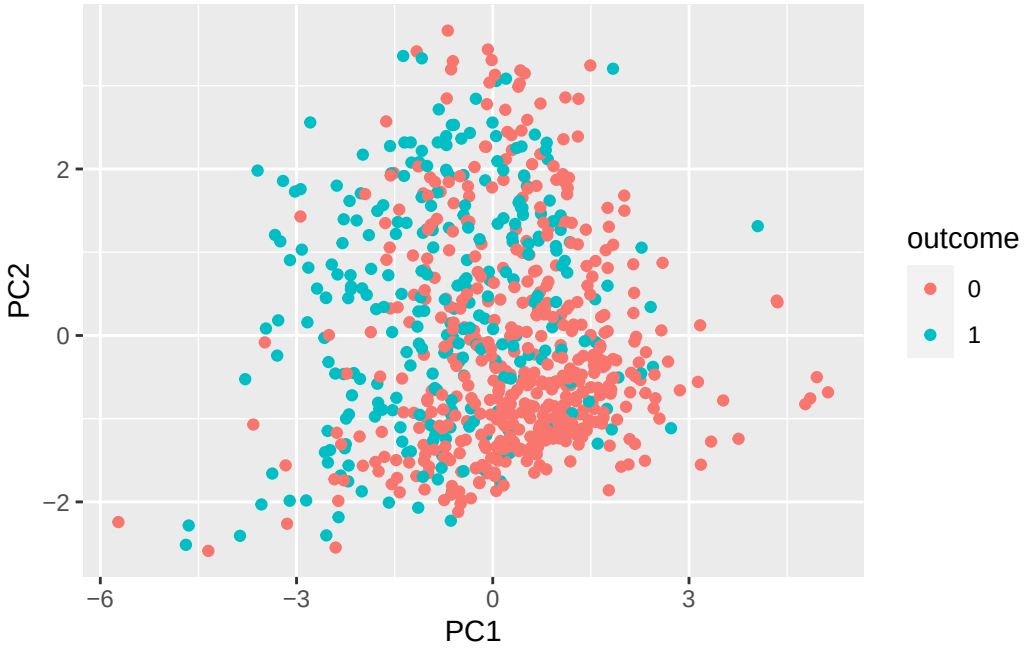
Pero de igual manera podemos escalar a ver si hay algun cambio...

- **Se escalan los datos "(scale. = T)" antes de realizar el PCA**

```
summary(datos)
```

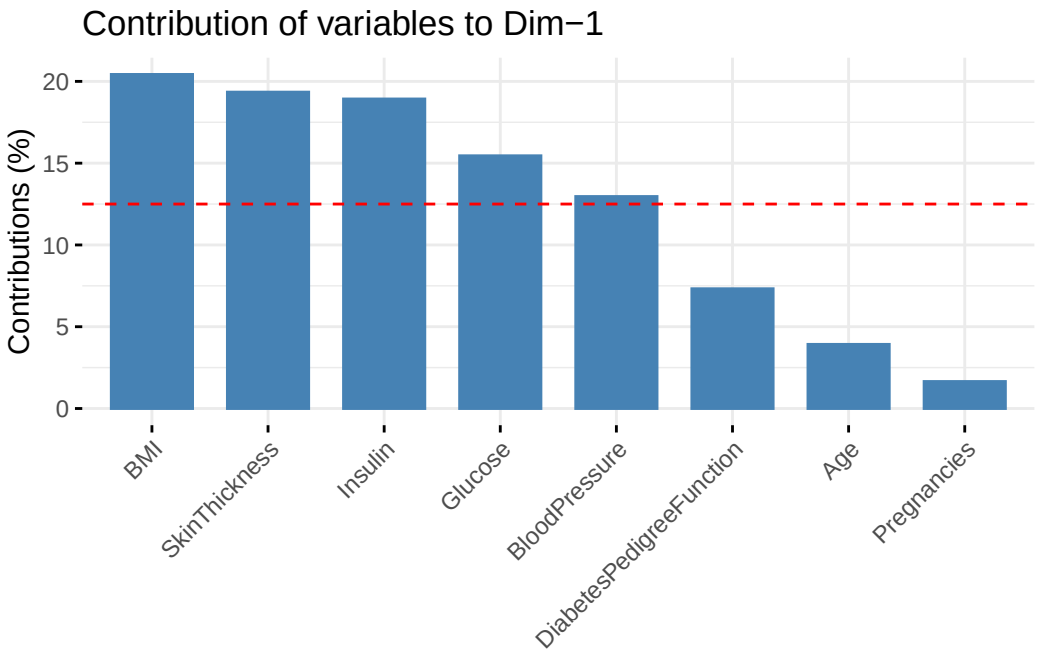
```
 Pregnancies        Glucose      BloodPressure    SkinThickness
Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
   Insulin           BMI        DiabetesPedigreeFunction      Age
Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
Outcome
0:500
1:268
```

```
pcx <- prcomp(datos[,1:n1],scale. = T) ## escalamos por la variablidad de los datos

plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()
```



- **El siguiente código influye en la estrcutura de componente principales y ayuda a identificar qué variables tienen un mayor impacto en la variabilidad capturada por el PCA.**
- **La función "fviz_contrib" calcula y visualiza las contribuciones de las variables a las componentes principales. Se puede realizar a través de gráfico de barras, cada barra representa la contribución relativa de una variable a cada componente principal.**

```
factoextra::fviz_contrib(pcx,"var")
```

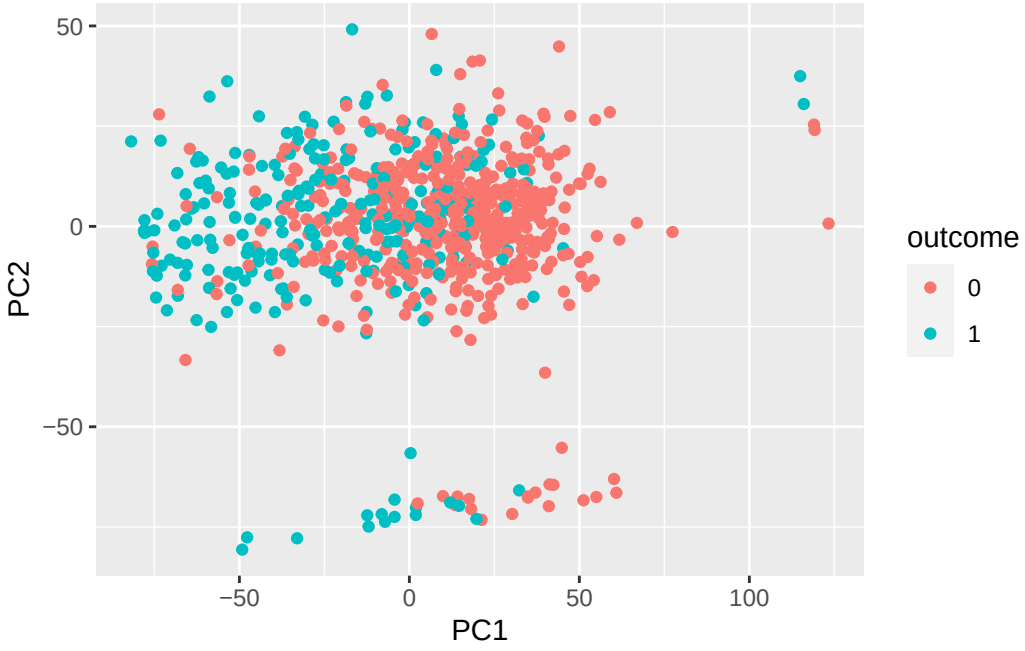


Al parecer es la insulina la que está dando problemas

- **El comando "grep" se usa para buscar las columnas del objeto "datos" que contengan la cadena de caracteres "insulin", ignorando la distinción entre mayúsculas y minúsculas "(ignore.case = TRUE)" Las columnas que coincidan se agregan al vector "w".**
- **Excluye las clumnas que contienen la cadena "Insulin" del conjunto de datos mediante la creación de la variable "w" que contiene las variables de las columnas que contienen "insulin" y el número total de columnas "(ncol(datos))".**
- **Se seleccionan todas las columnas excepto las indicadas en la variable "w" para realizar el PCA sin escalar los datos nuevamente, utilizando datos "[, -w]" en la función "prcomp()".**
- **Se visualiza las dos primeras componentes principales en un gráfico de dispersión, donde los puntos se colorearán según los valores de la variable "Outcome"**

```
## indices a quitar
w <- c(grep("insulin",ignore.case = T,colnames(datos)),ncol(datos))
pcx <- prcomp(datos[,-w],scale. = F) ## escalamos por la variablidad de los datos

plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)
```

```
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()
```



De hecho la insulina, tenía un aspecto raro, como sesgado, ver gráficos de arriba. Vamos a transformala...

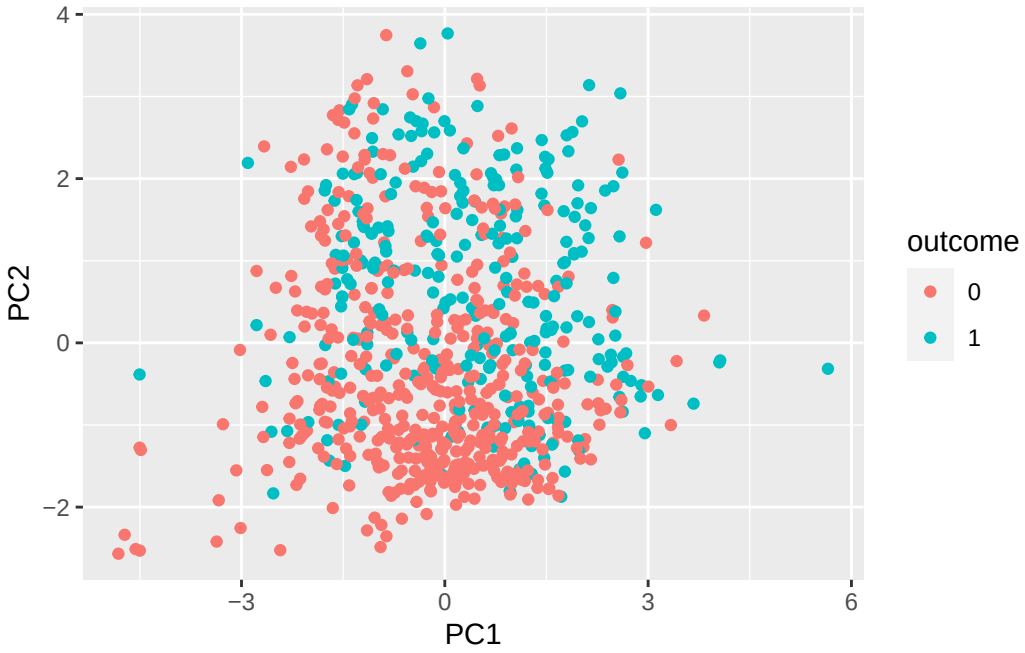
- **Se utiliza el comando "datos$Insulin <- log(datos$Insulin+0.05)" calcula el logaritmo natural (base e) de la variable "Insulin"y lo agrega a la misma columna . Se agrega 0.05 al valor original antes de aplicar el logaritmo para evitar problemas/erorres con valores cercanos a cero.**

```
datos$Insulin  <- log(datos$Insulin+0.05)

summary(datos)
```

```
  Pregnancies        Glucose      BloodPressure    SkinThickness
Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
```

```
   Insulin             BMI         DiabetesPedigreeFunction      Age
Min.   :-2.996   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
1st Qu.:-2.996   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
Median : 3.418   Median :32.00   Median :0.3725           Median :29.00
Mean   : 1.008   Mean   :31.99   Mean   :0.4719           Mean   :33.24
3rd Qu.: 4.847   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
Max.   : 6.741   Max.   :67.10   Max.   :2.4200           Max.   :81.00
Outcome
0:500
1:268
```

```r
pcx <- prcomp(datos[,1:n1],scale. = T) ## escalamos por la variablidad de los datos

plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()
```



Cambia ! Esto significa que no hemos quitado la infromacion de la insulina, solamente lo hemos transformado.

Es decir, cambia si transformamos los datos…a partir de esto, podemos realizar de nuevo pruebas de diferencia de medianas, pero ahora lo veremos condensado..

- **Cargamos nuevamente los daots y se convierte en factor y escalamos los datos.**
- **"scale" es una función genérica cuyo método por defecto centra y/o escala las columnas de una matriz numérica.**

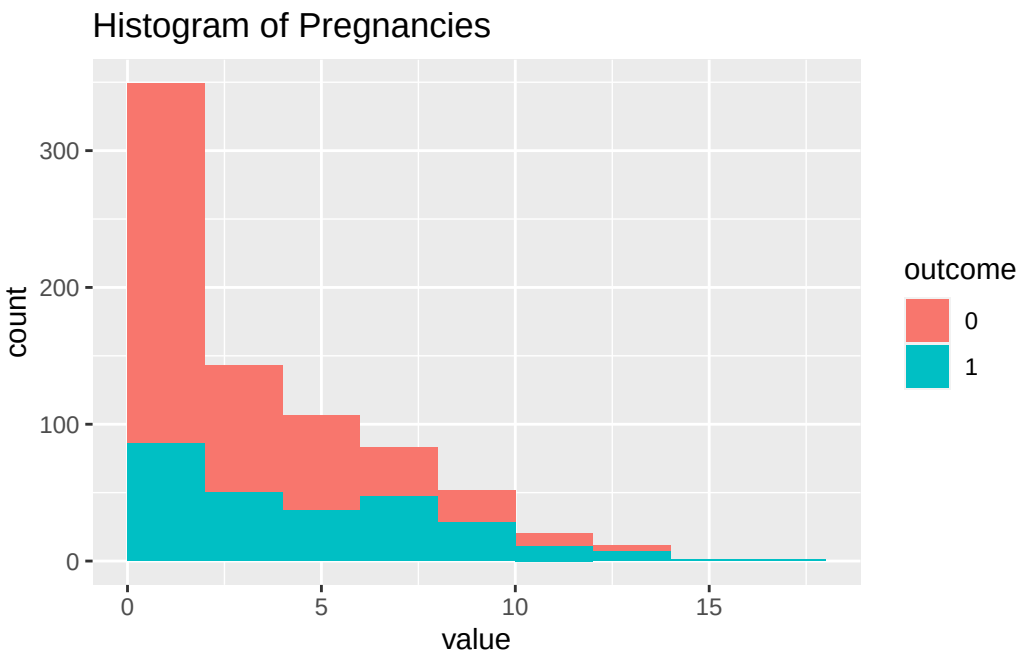
```
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <- as.factor(datos$Outcome)
datsc <- scale(datos[,-ncol(datos)])
```

Veamos las distribuciones de nuevo….

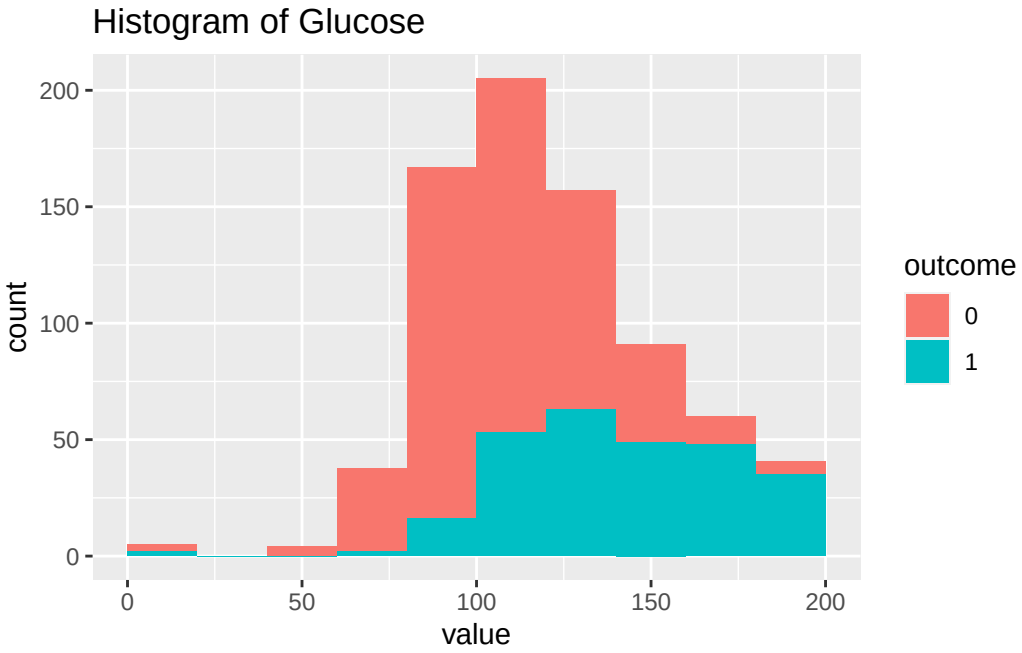
- **Haciendo uso de un bucle for para visualizar las distribuciones. Para cada variable, se produce un histograma y se mantiene en una lista de l.plots**

```
l.plots <- vector("list",length = ncol(datos)-1)
n1 <- ncol(datos) -1
for(j in 1:n1){

  h <-hist(datos[,j],plot = F)
  datos.tmp <- data.frame(value=datos[,j],outcome=datos$Outcome)
  p1 <- ggplot(datos.tmp,aes(value,fill=outcome))+geom_histogram(breaks=h$breaks) + ggtitl

  l.plots[[j]] <- p1
}
l.plots
```

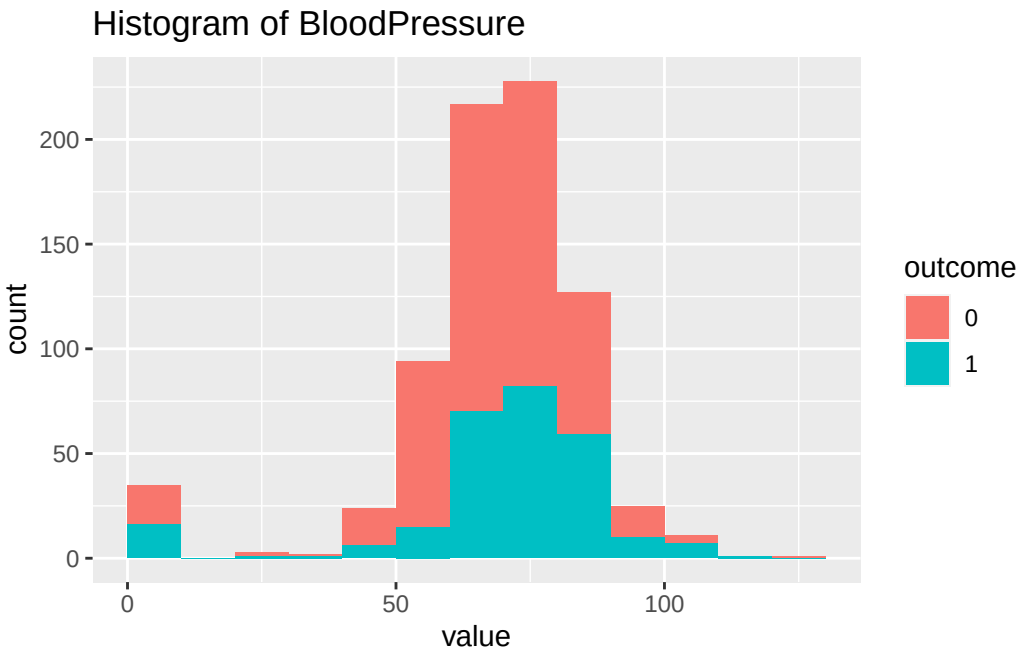
```
[[1]]
```


Histogram of Pregnancies
count
value
outcome
0
1
0
100
200
300
0
5
10
15

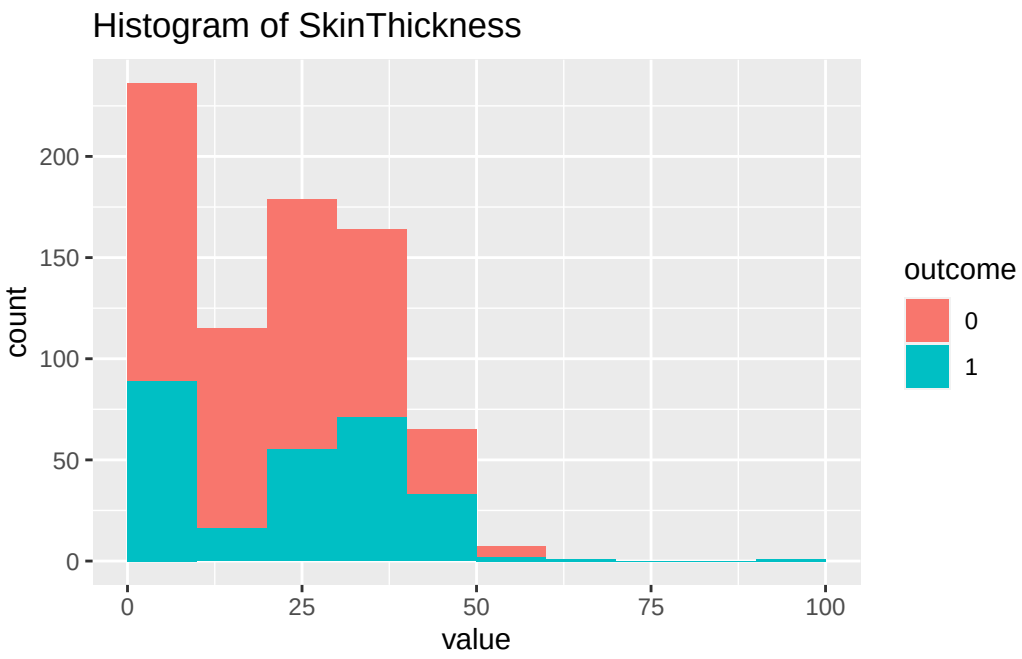
```
[[2]]
```


Histogram of Glucose
count
value
outcome
0
1
0
50
100
150
200
0
50
100
150
200

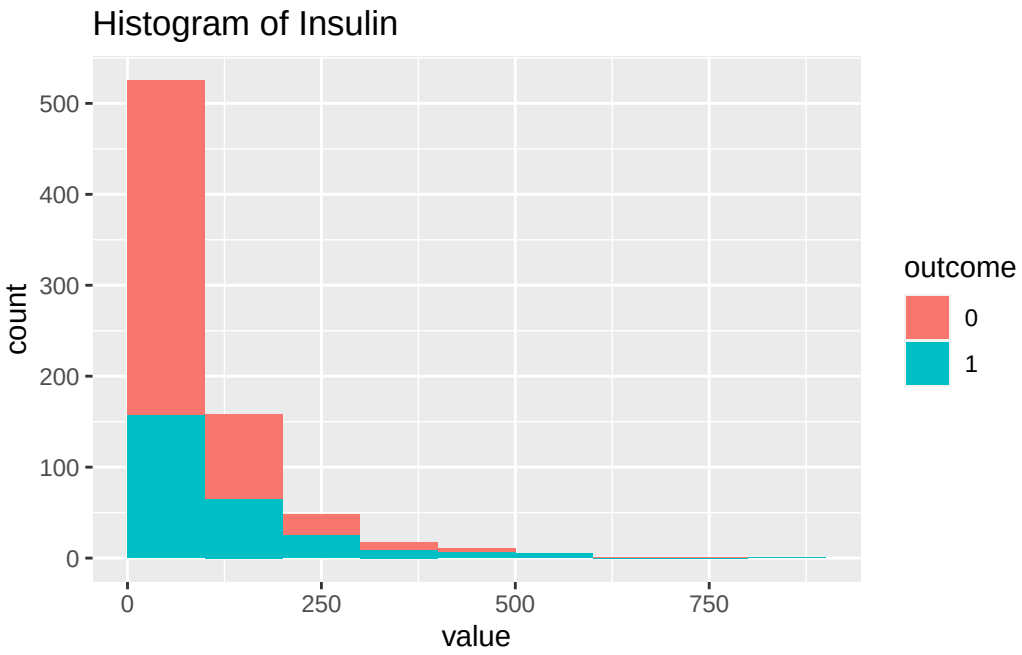
```
[[3]]
```


Histogram of BloodPressure
count
200
150
100
50
0
0
50
100
value
outcome
0
1

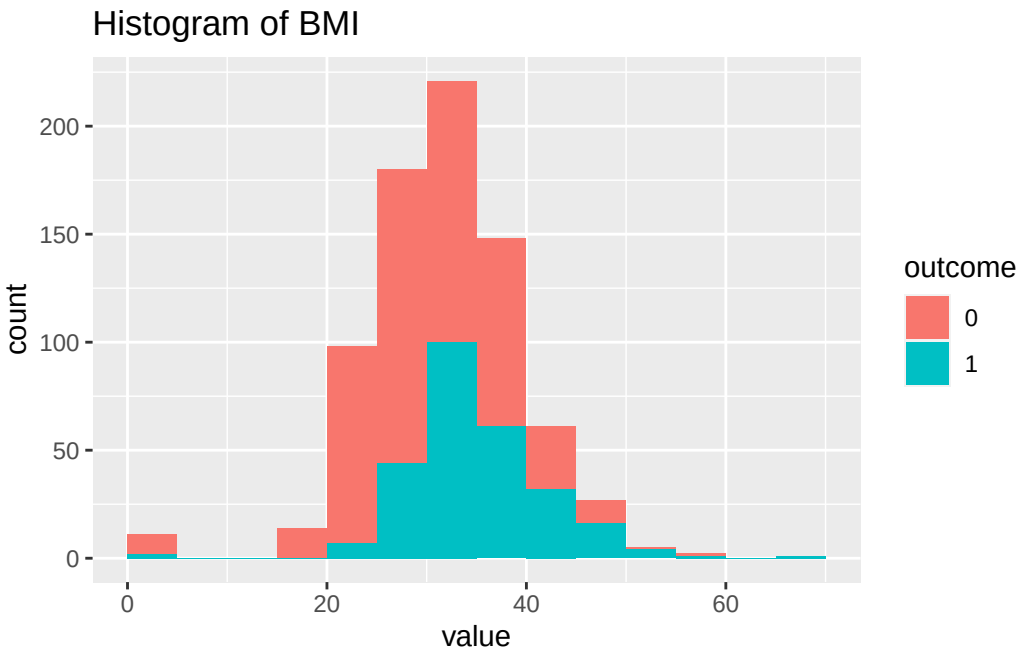
```
[[4]]
```


Histogram of SkinThickness
count
value
outcome
0
1
0
50
100
150
200
0
25
50
75
100

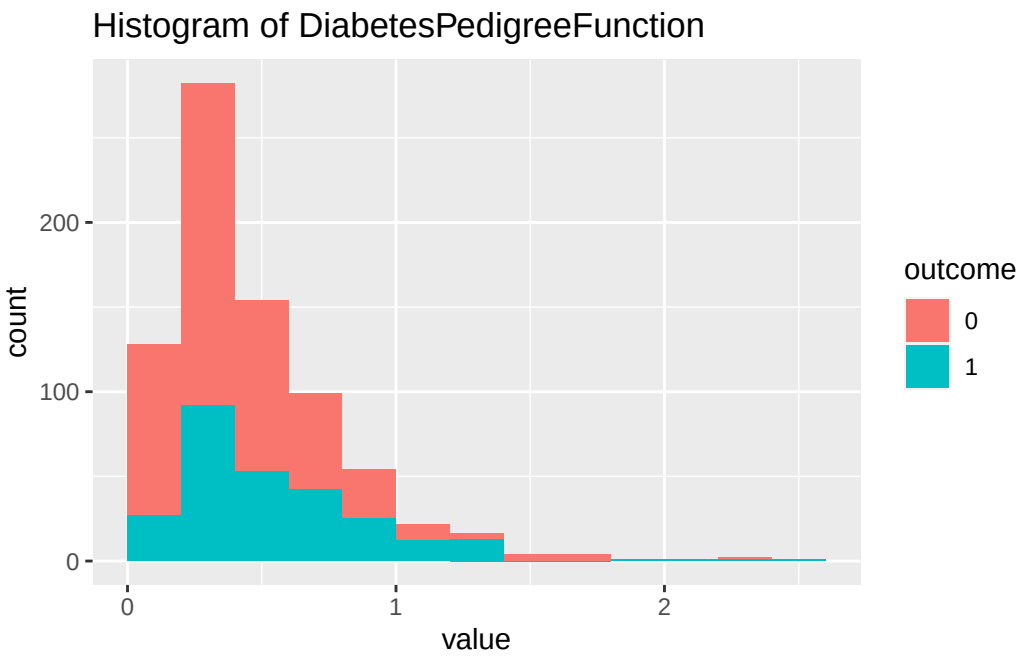
```
[[5]]
```


Histogram of Insulin
count
value
outcome
0
1
0
100
200
300
400
500
0
250
500
750

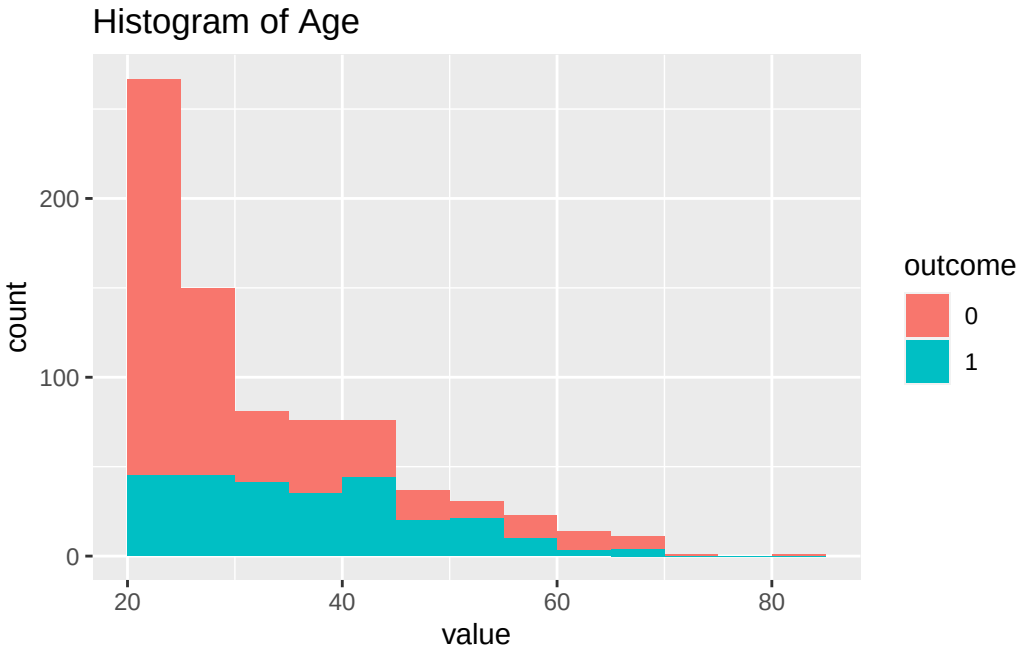
[[6]]


Histogram of BMI
count
200
150
100
50
0
0
20
40
60
value
outcome
0
1

[[7]]


Histogram of DiabetesPedigreeFunction
count
200
100
0
0
1
2
value
outcome
0
1

```
[[8]]
```

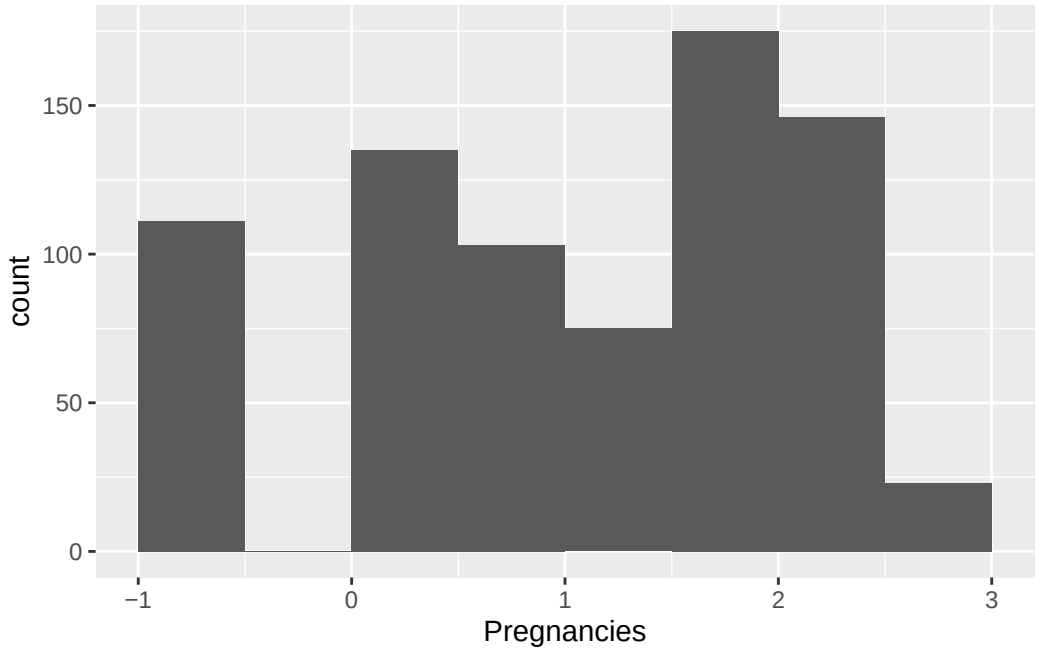

Histogram of Age
count
200
100
0
20
40
60
80
value
outcome
0
1

Curioso, los valores la insulina, han cambiado por la transformación en valor mas no la distribución, vamos a hacer unos arrelgos...

Al parecer la preñanza esta ligada a una esgala logaritmica de 2 Esto es otra cosa...

- **Se realiza una transformación logarítmica en la variable "Pregnancies" agregando 0.5 a cada valor antes de aplicar el logaritmo.**
- **Ademas se realiza un diagrama de barras.**

```
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <- as.factor(datos$Outcome)
datos$Pregnancies  <- log(datos$Pregnancies+0.5)
ggplot(datos,aes(Pregnancies))+geom_histogram(breaks = hist(datos$Pregnancies,plot=F)$brea
```

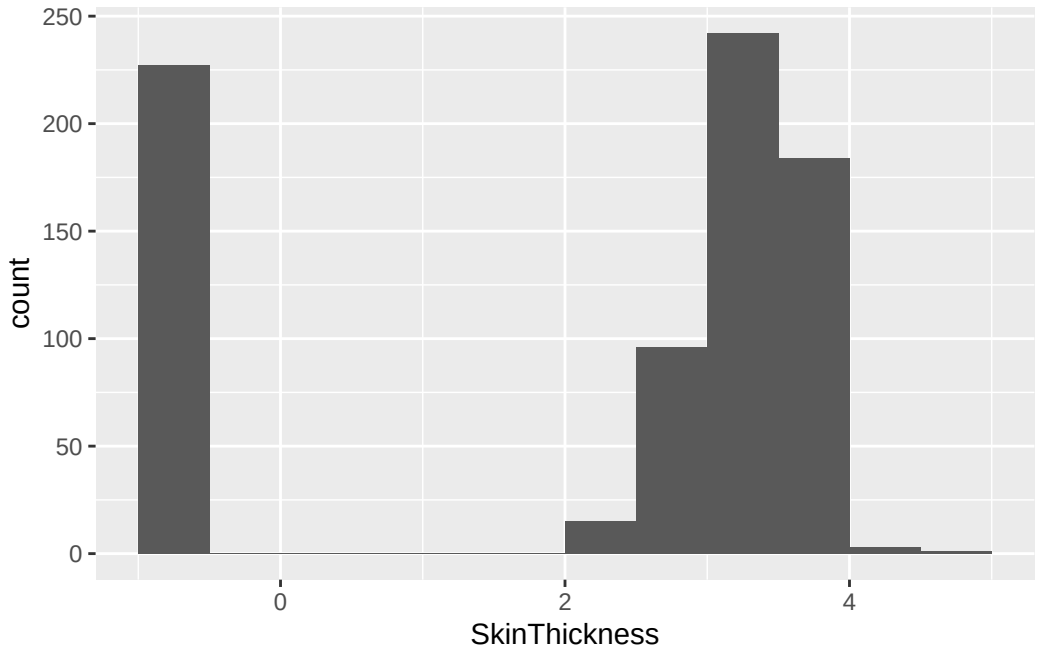


Realizaremos lo mismo con la grosura de la piel

- **Se carga nuevamente los datos, convertimos en factor, además seleccionamos los datos que se refieren a la grosura de la piel "SkinThickness".**

```
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <- as.factor(datos$Outcome)
datos$SkinThickness  <- log(datos$SkinThickness+0.5)
```

```
ggplot(datos,aes(SkinThickness))+geom_histogram(breaks = hist(datos$SkinThickness,plot=F)$
```

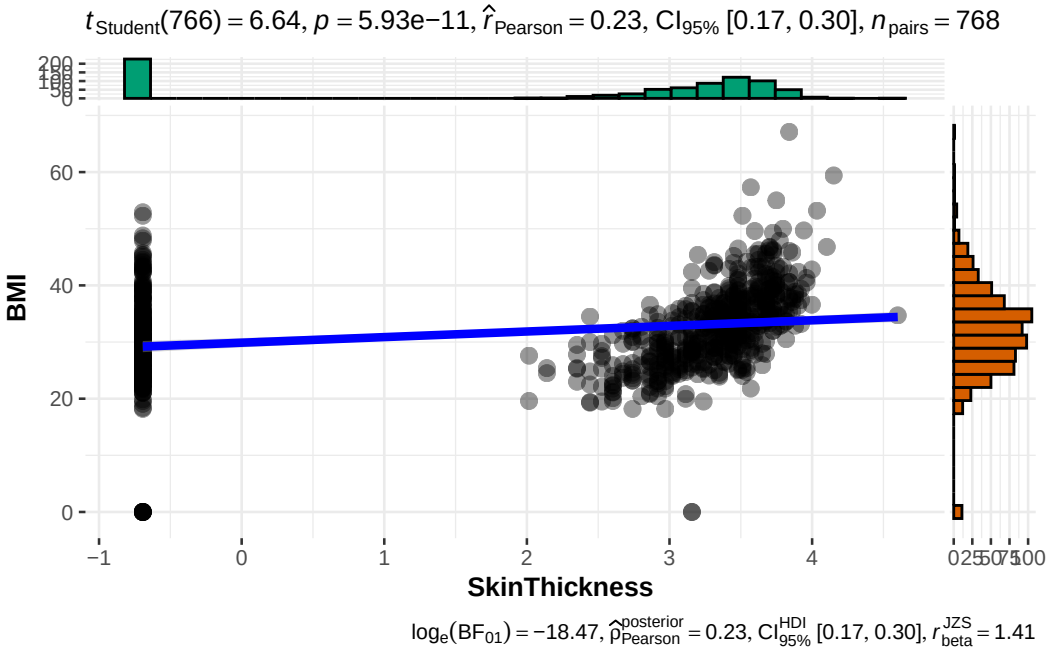


Tenemos algo raro, lo más posible sea por la obesidad...

- **La posición del eje x de cada punto en el gráfico corresponde al valor "SkinThickness", mientras que la posición del eje y al valor "BMI" para cada observación en el conjunto de datos.**

```
ggscatterstats(datos,SkinThickness,BMI)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Curioso ! al parecer los datos tienen valores nulos, los cuales solo están en las otras variables que no sean pregnancies. Vamos a quitarlos...

- **Para quitar los valores que no sean pregnacies la función "apply()" junto con "ifelse()" reemplazan los valores nulos (0) en todas las columnas, excepto en las columnas "Pregnancies" y "Outcome", con NA.**

```
datos <- read.csv("./datos/diabetes.csv")
datos[,-c(1,9)] <- apply(datos[,-c(1,9)],2,function(x) ifelse(x==0,NA,x))

datos$Outcome <- as.factor(datos$Outcome)
```

vamos a quitar estos valores

- **El comando "complete.cases()" identifica las filas en el conjunto de datos donde no hay valores faltantes en ninguna columna.**

```
datos <- datos[complete.cases(datos),]
```

Se redujo el data set a 392 observaciones...

- **El comando "table()" permite obtener la frecuencia de cada nivel en la variable "Outcome" del conjunto de datos actualizado**

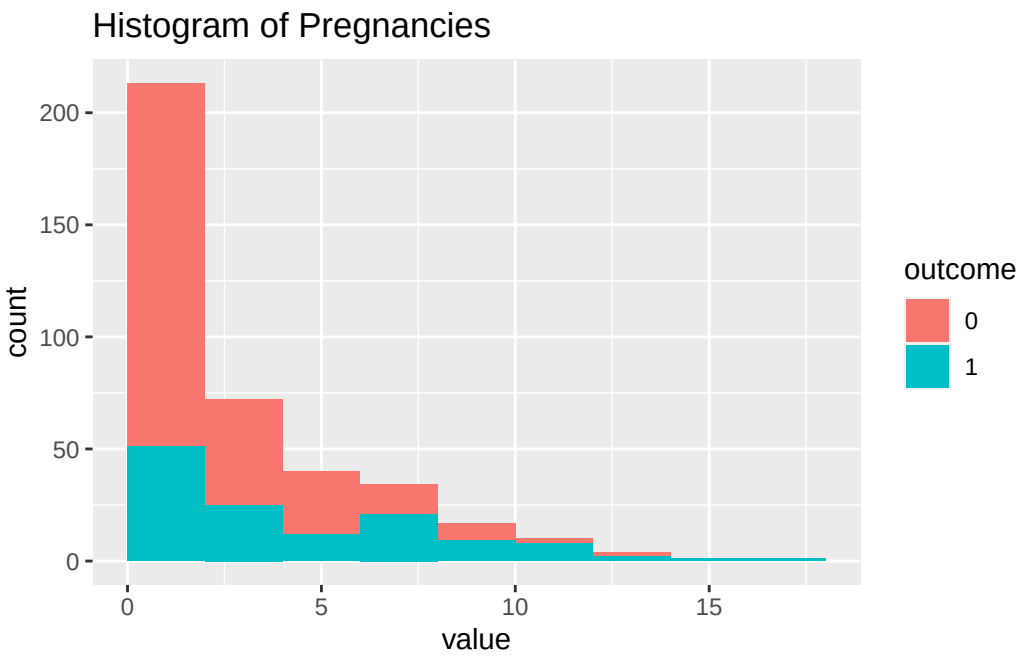
```
table(datos$Outcome)
```

```
  0   1
262 130
```

- **El comando "l . plots" genera una lista vacía con una longitud igual al número de columnas de datos; sin embargo, haremos lo mismo restándole 1.**
- **Las columnas "value" y "outcome" se agregan a un nuevo "data.frame" llamado "datos.tmp".**

```
l.plots <- vector("list",length = ncol(datos)-1)
n1 <- ncol(datos) -1
for(j in 1:n1){

  h <-hist(datos[,j],plot = F)
  datos.tmp <- data.frame(value=datos[,j],outcome=datos$Outcome)
  p1 <- ggplot(datos.tmp,aes(value,fill=outcome))+geom_histogram(breaks=h$breaks) + ggtitl

  l.plots[[j]] <- p1
}
l.plots
```

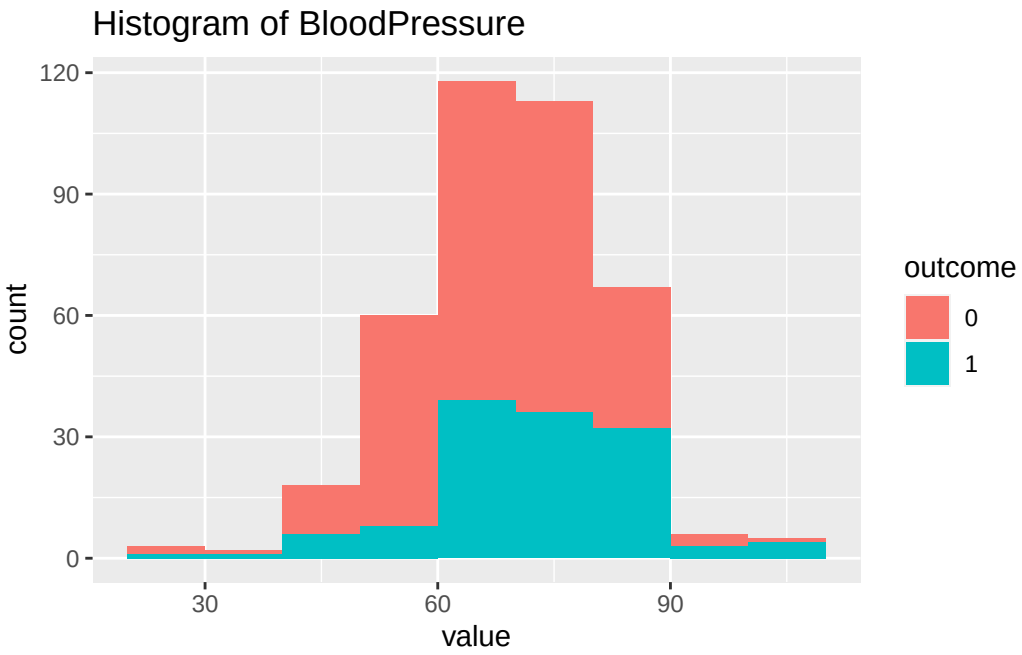
```
[[1]]
```


Histogram of Pregnancies
count
value
outcome
0
1
0
50
100
150
200
0
5
10
15

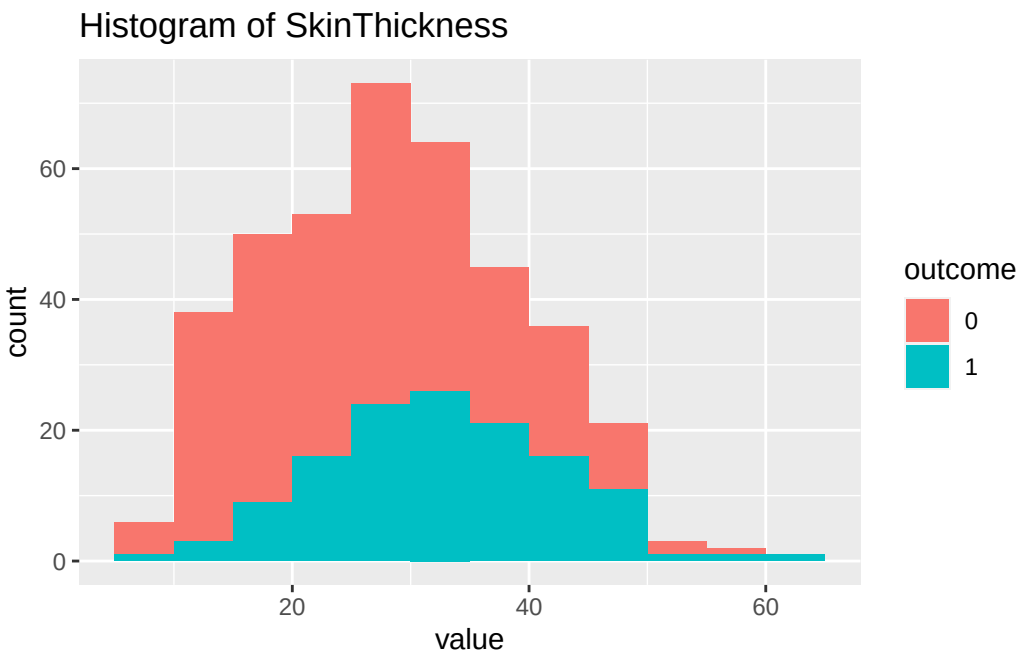
```
[[2]]
```


Histogram of Glucose
count
value
outcome
0
1
0
25
50
75
100
50
100
150
200

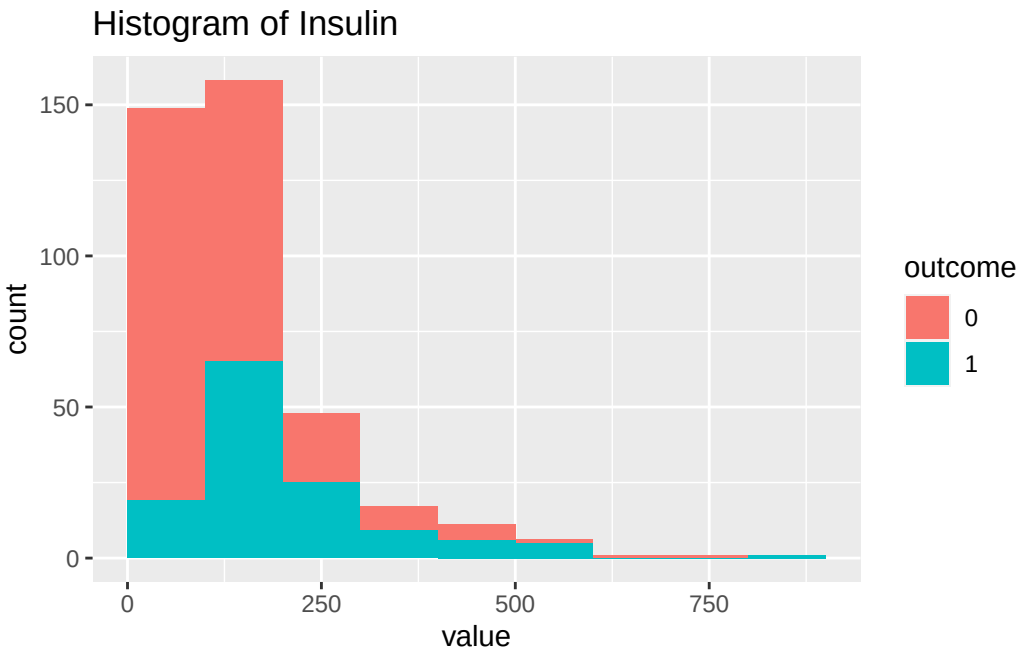
```
[[3]]
```


Histogram of BloodPressure
120
90
60
30
0
count
30
60
90
value
outcome
0
1

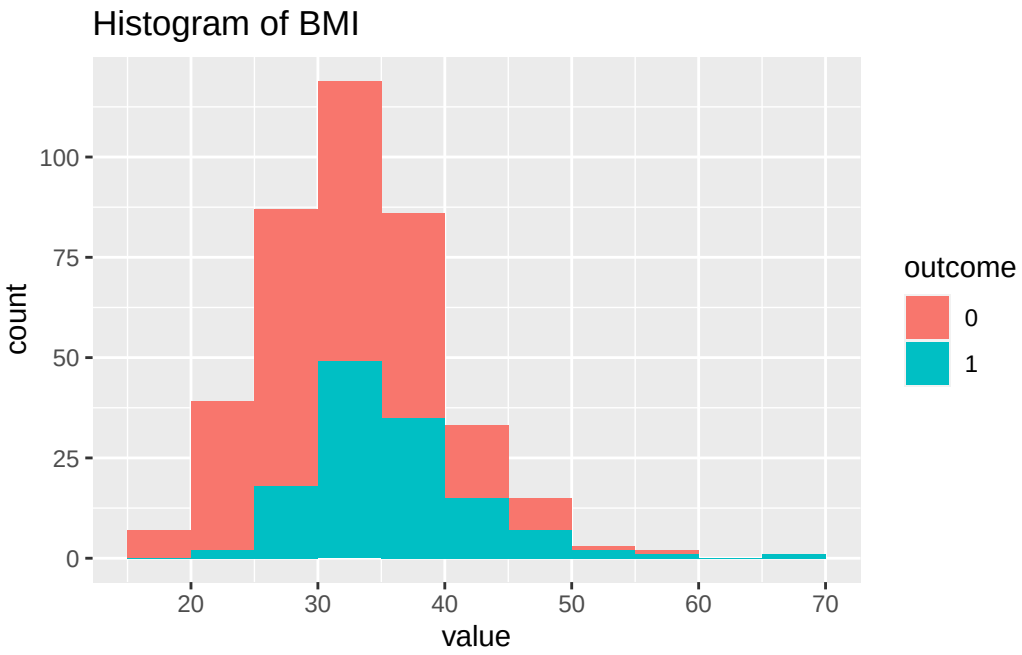
```
[[4]]
```


Histogram of SkinThickness
count
value
outcome
0
1
0
20
40
60
0
20
40
60

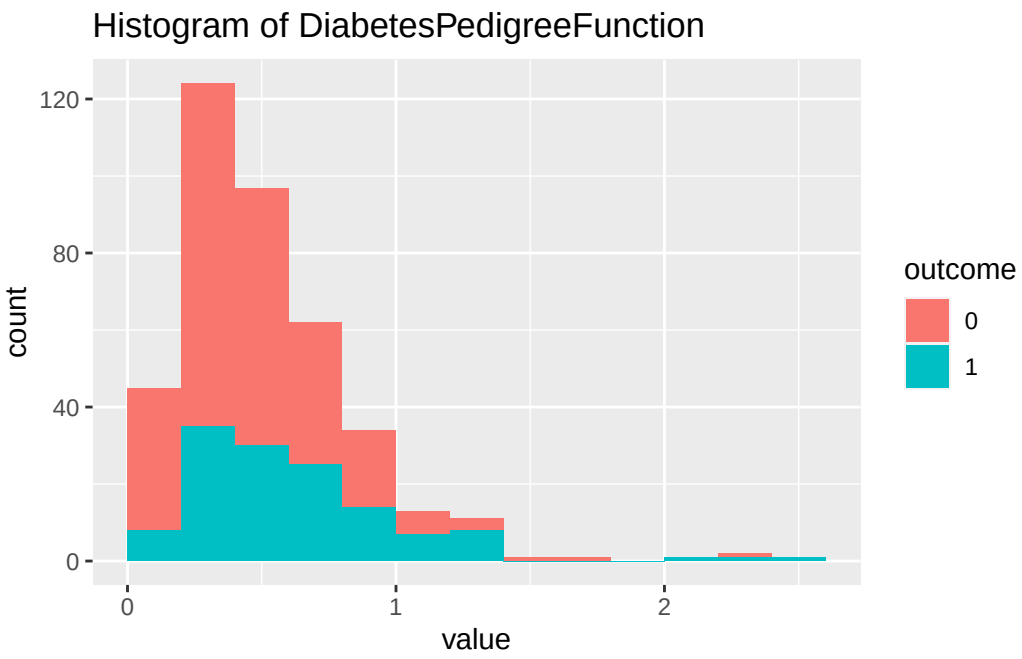
```
[[5]]
```


Histogram of Insulin
count
value
outcome
0
1
0
50
100
150
0
250
500
750

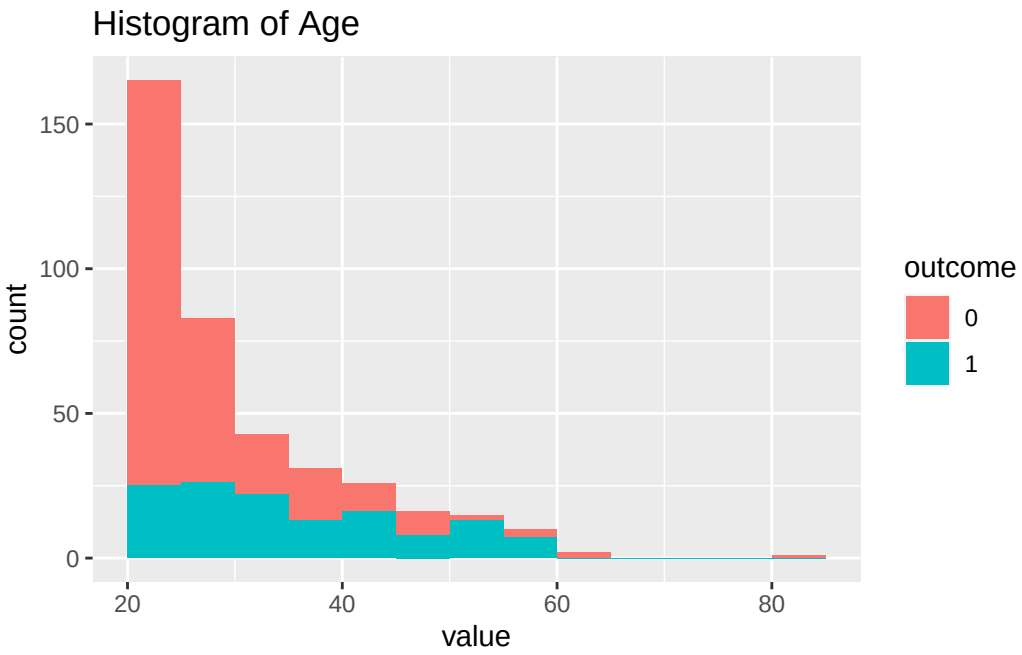
```
[[6]]
```


Histogram of BMI
count
100
75
50
25
0
20
30
40
50
60
70
value
outcome
0
1

```
[[7]]
```


Histogram of DiabetesPedigreeFunction
count
value
outcome
0
1
120
80
40
0
0
1
2

```
[[8]]
```

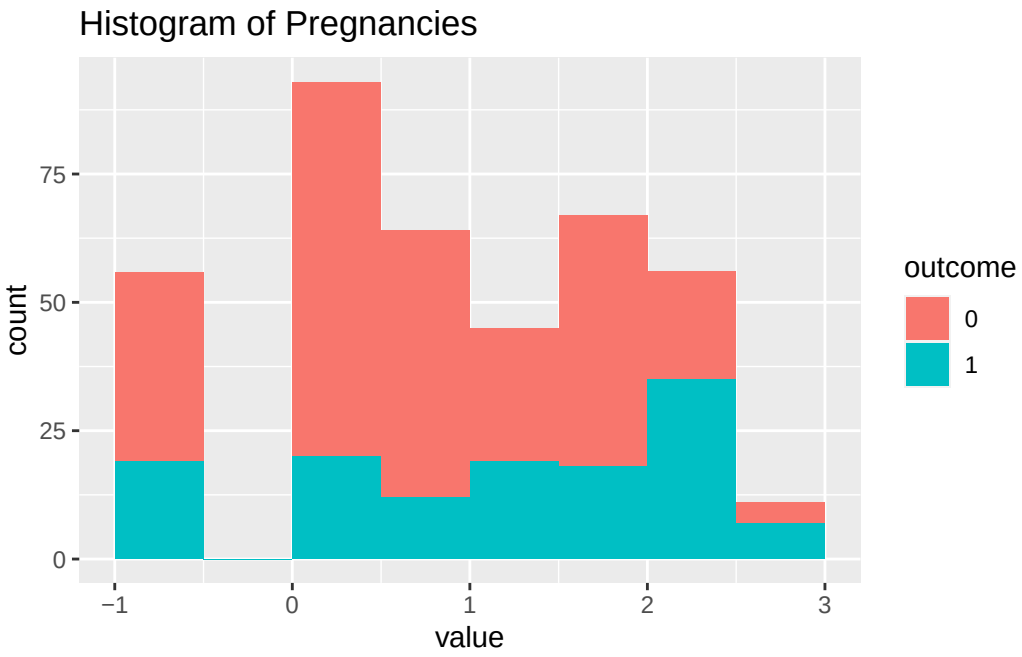

Histogram of Age
count
value
outcome
0
1
150
100
50
0
20
40
60
80

Ahora si podemos realizar las transfomraciones

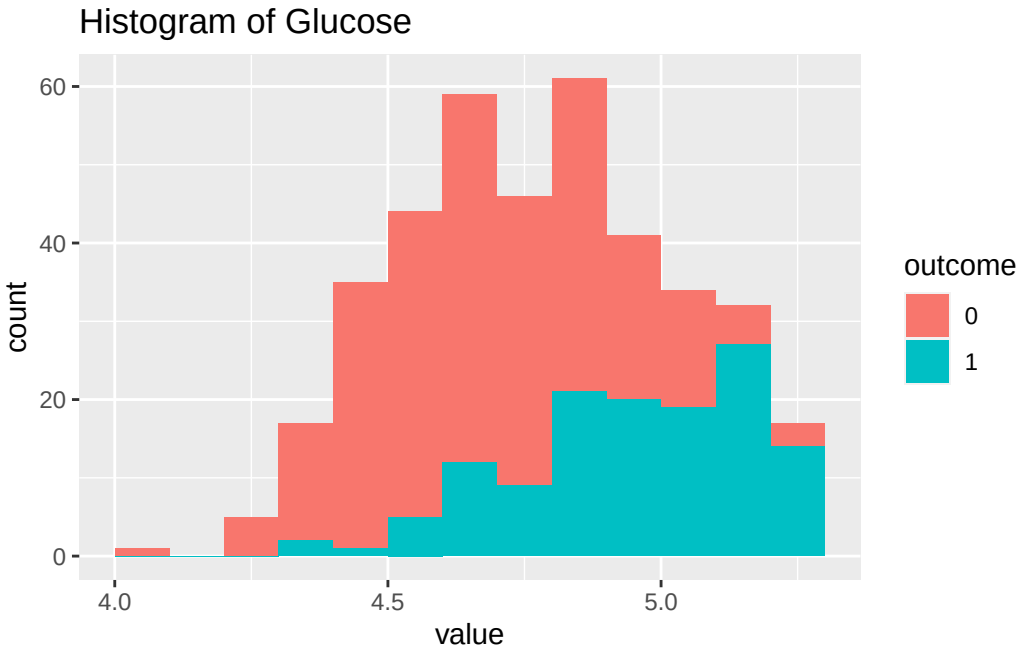
- **Cargar el archivo "diabetes.csv".**
- **Usemos 'as.factor()" para convertir la columna "Outcome" en una variable categórica o factorial, una función que factoriza.**
- **La aplicación de las funciones logarítmicas "Insulin", "Pregnacies" y "DiabetesPedigreeFunction" creará una transformación logarítmica.**
- **La función de raíz cuadrada de la columna "SkinThickness". Las columnas "Glucose" y "Age" están sujetas a la función logarítmica de base 2.**

```
datos <- read.csv("./datos/diabetes.csv")
datos[,-c(1,9)] <- apply(datos[,-c(1,9)],2,function(x) ifelse(x==0,NA,x))
datos <- datos[complete.cases(datos),]

datos$Outcome <- as.factor(datos$Outcome)
datos$Insulin <- log(datos$Insulin)
datos$Pregnancies <- log(datos$Pregnancies+0.5)
datos$DiabetesPedigreeFunction <- log(datos$DiabetesPedigreeFunction)

datos$SkinThickness <- sqrt((datos$SkinThickness))
datos$Glucose <- log(datos$Glucose)
datos$Age <-log2(datos$Age)
l.plots <- vector("list",length = ncol(datos)-1)
n1 <- ncol(datos) -1
for(j in 1:n1){

  h <-hist(datos[,j],plot = F)
  datos.tmp <- data.frame(value=datos[,j],outcome=datos$Outcome)
  p1 <- ggplot(datos.tmp,aes(value,fill=outcome))+geom_histogram(breaks=h$breaks) + ggtitl

  l.plots[[j]] <- p1
}
l.plots
```

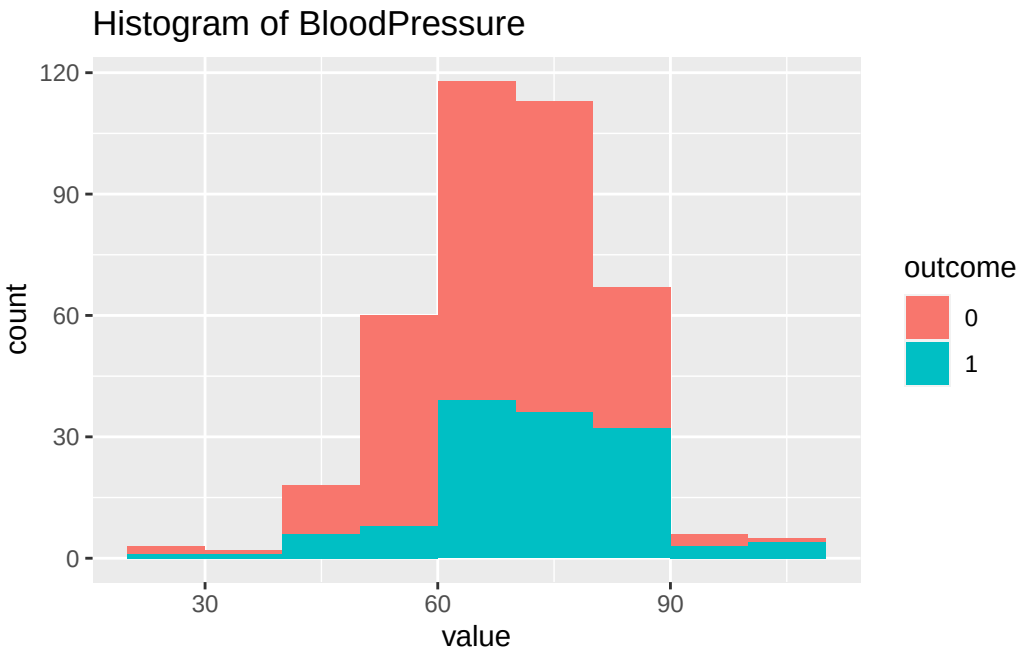
```
[[1]]
```


Histogram of Pregnancies
count
75
50
25
0
−1
0
1
2
3
value
outcome
0
1

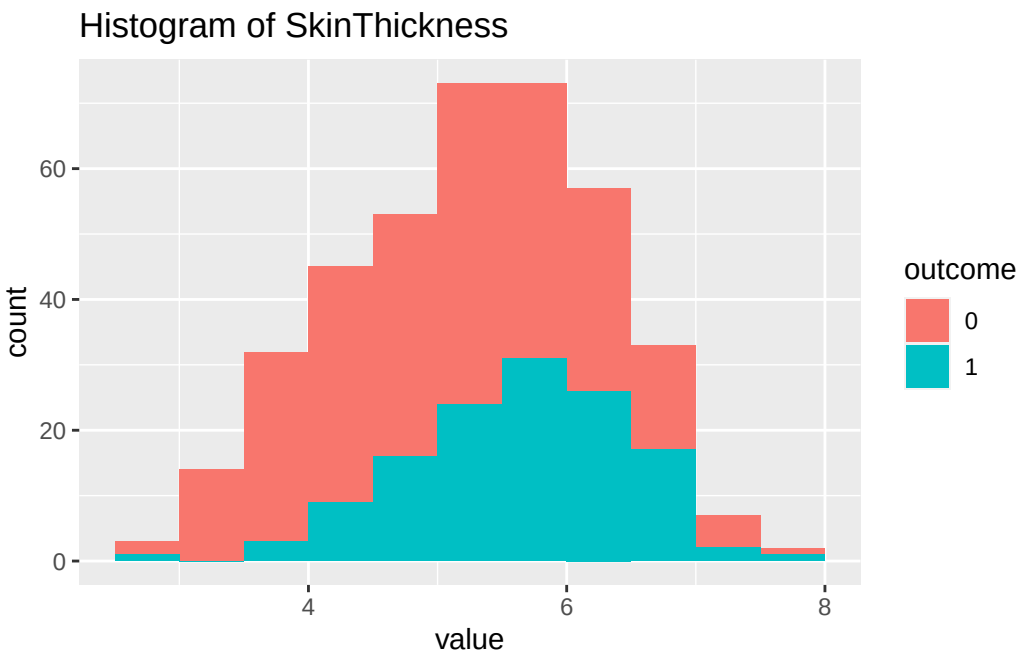
```
[[2]]
```


Histogram of Glucose
count
60
40
20
0
4.0
4.5
5.0
value
outcome
0
1

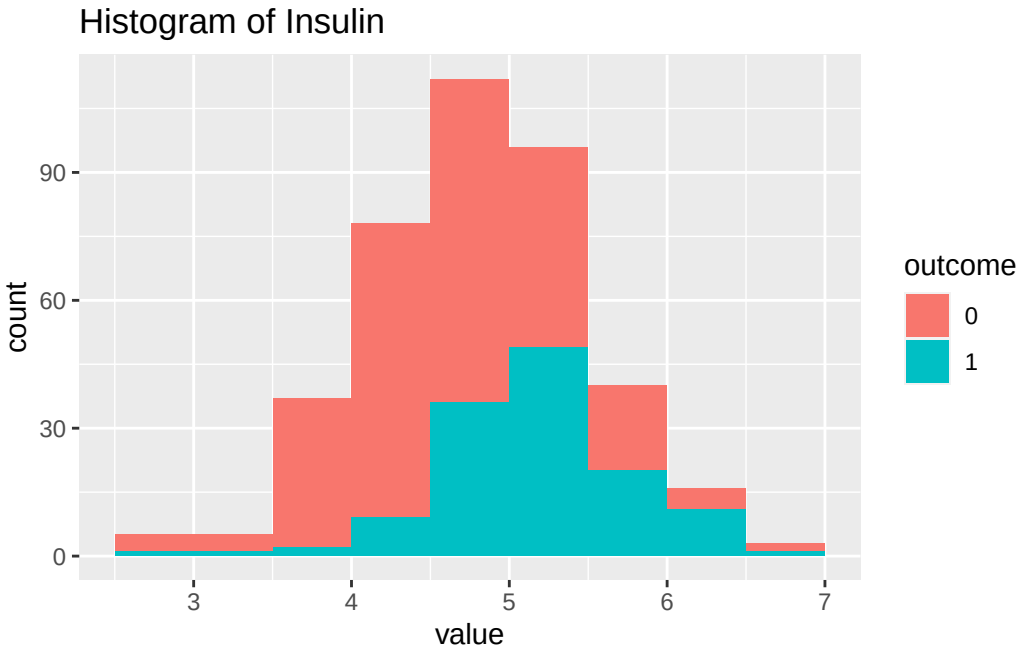
```
[[3]]
```


Histogram of BloodPressure
120
90
60
30
0
count
30
60
90
value
outcome
0
1

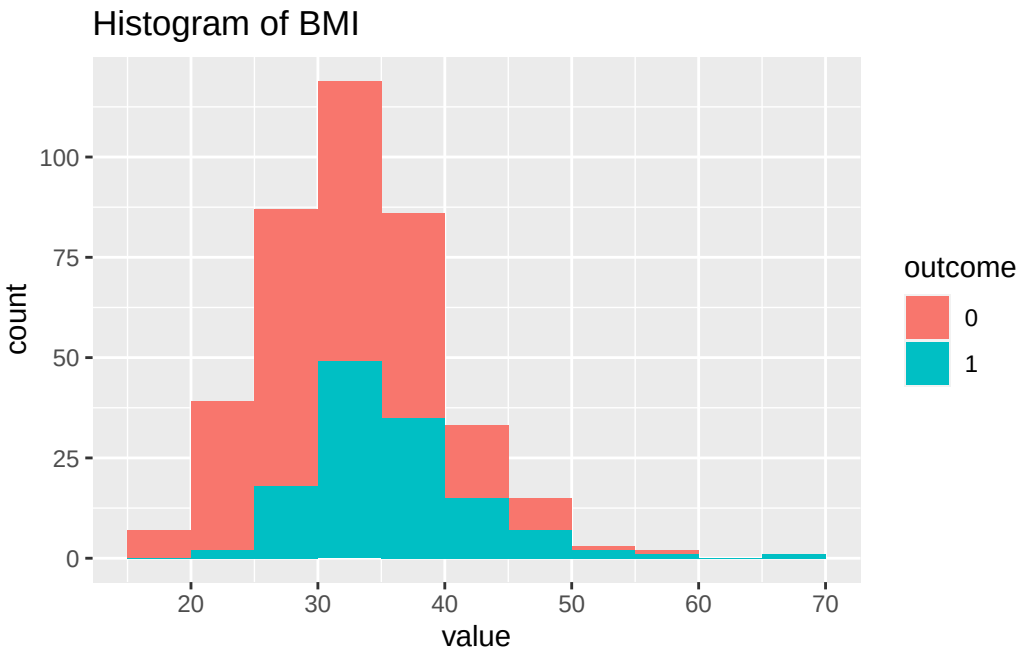
```
[[4]]
```


Histogram of SkinThickness
count
value
outcome
0
1
0
20
40
60
4
6
8

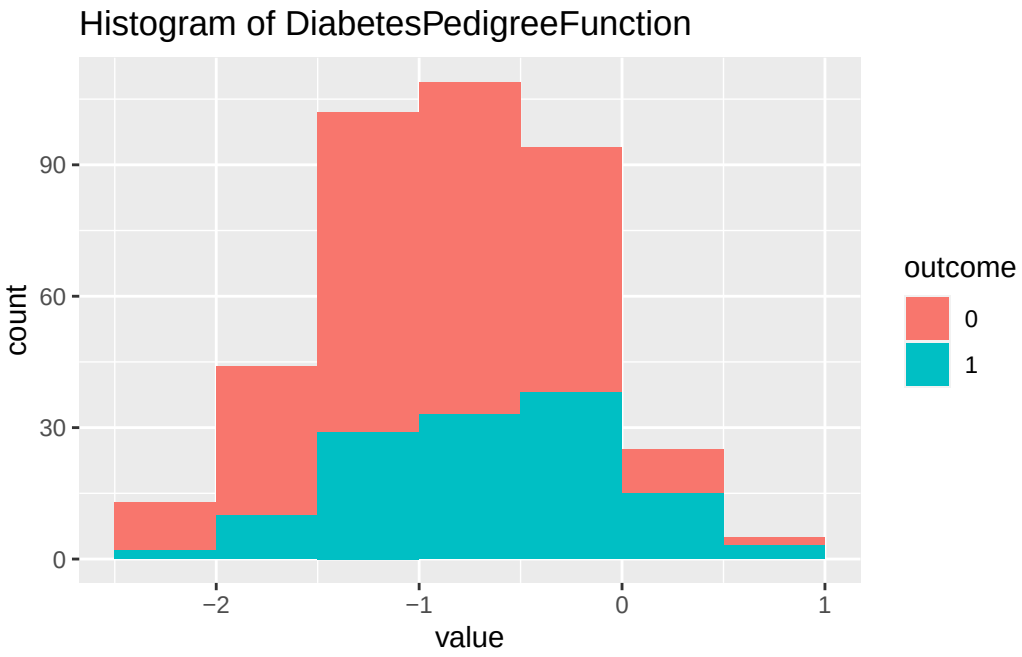
```
[[5]]
```


Histogram of Insulin
count
value
outcome
0
1
0
30
60
90
3
4
5
6
7

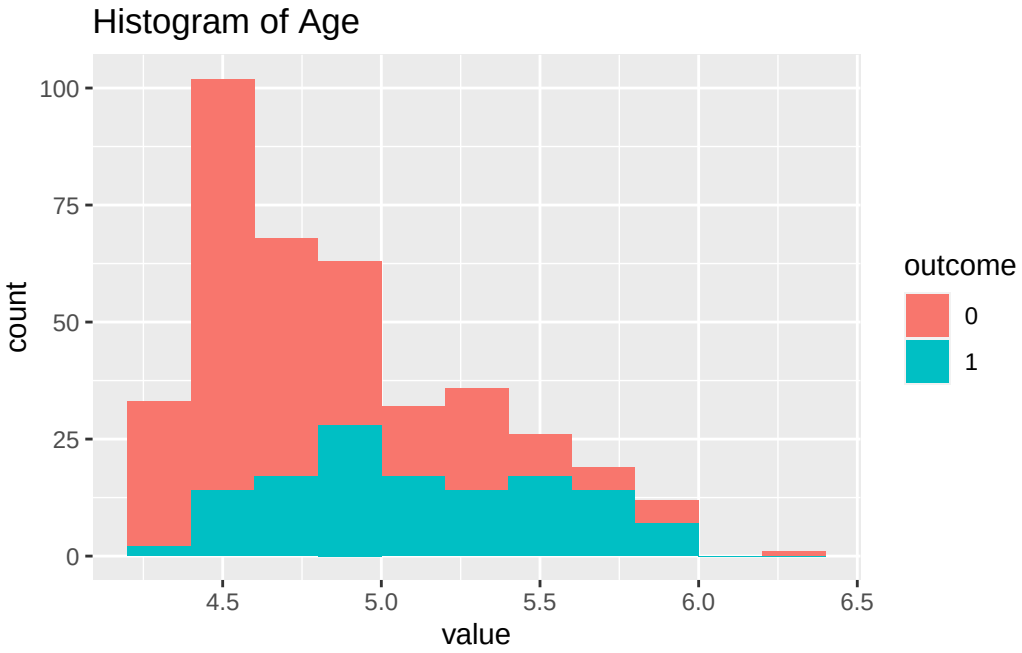
```
[[6]]
```


Histogram of BMI
count
100
75
50
25
0
20
30
40
50
60
70
value
outcome
0
1

```
[[7]]
```


Histogram of DiabetesPedigreeFunction
count
90
60
30
0
−2
−1
0
1
value
outcome
0
1

```
[[8]]
```

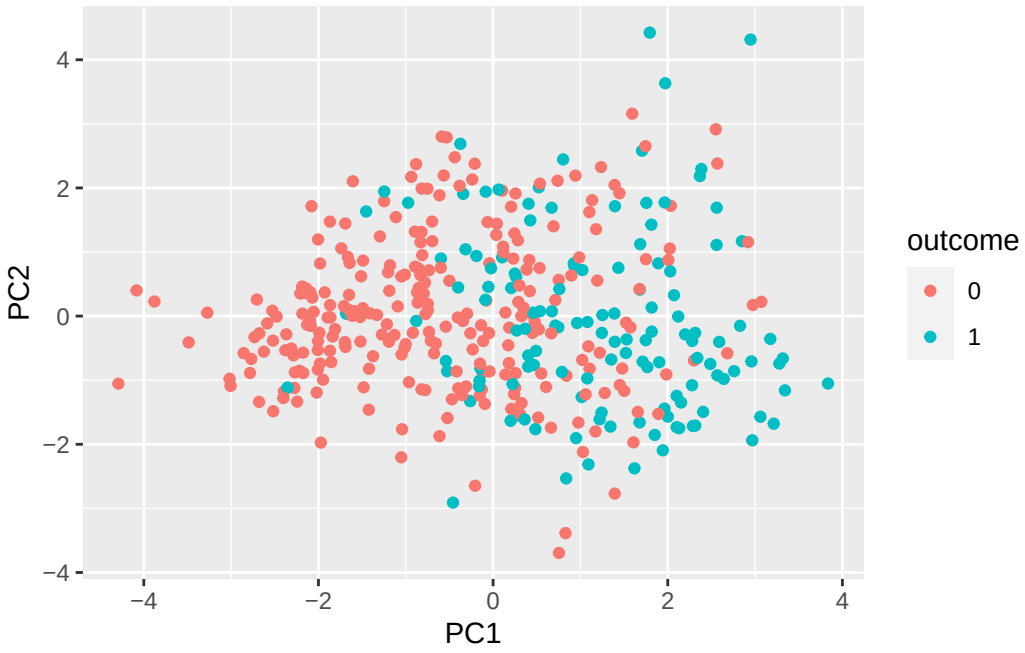

Histogram of Age
count
100
75
50
25
0
4.5
5.0
5.5
6.0
6.5
value
outcome
0
1

Con las anteriores transformaciones vamos a realizar el PCA de nuevo.

```
summary(datos)
```

```
 Pregnancies         Glucose      BloodPressure    SkinThickness
Min.   :-0.6931   Min.   :4.025   Min.   : 24.00   Min.   :2.646
1st Qu.: 0.4055   1st Qu.:4.595   1st Qu.: 62.00   1st Qu.:4.583
Median : 0.9163   Median :4.779   Median : 70.00   Median :5.385
Mean   : 0.9590   Mean   :4.778   Mean   : 70.66   Mean   :5.305
3rd Qu.: 1.7047   3rd Qu.:4.963   3rd Qu.: 78.00   3rd Qu.:6.083
Max.   : 2.8622   Max.   :5.288   Max.   :110.00   Max.   :7.937
   Insulin           BMI        DiabetesPedigreeFunction      Age
Min.   :2.639   Min.   :18.20   Min.   :-2.4651          Min.   :4.392
1st Qu.:4.341   1st Qu.:28.40   1st Qu.:-1.3103          1st Qu.:4.524
Median :4.832   Median :33.20   Median :-0.7996          Median :4.755
Mean   :4.813   Mean   :33.09   Mean   :-0.8391          Mean   :4.882
3rd Qu.:5.247   3rd Qu.:37.10   3rd Qu.:-0.3754          3rd Qu.:5.170
Max.   :6.741   Max.   :67.10   Max.   : 0.8838          Max.   :6.340
Outcome
0:262
1:130
```

```
pcx <- prcomp(datos[,1:n1],scale. = T) ## escalamos por la variablidad de los datos

plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()
```



Ahora vamos a realizar las pruebas de medianas

```
p.norm <- apply(apply(scale(datos[,1:n1]),
            2,
            function(x) summary(lm(x~datos$Outcome))$residuals),
      2,
      shapiro.test)

p.norm
```

```
$Pregnancies

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.95146, p-value = 4.684e-10


$Glucose

    Shapiro-Wilk normality test
```

```
data:  newX[, i]
W = 0.9958, p-value = 0.3813


$BloodPressure

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.99011, p-value = 0.009686


$SkinThickness

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.99384, p-value = 0.1123


$Insulin

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.99054, p-value = 0.0128


$BMI

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.97122, p-value = 5.374e-07


$DiabetesPedigreeFunction

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.99456, p-value = 0.1796
```

```
$Age

    Shapiro-Wilk normality test

data:  newX[, i]
W = 0.93053, p-value = 1.561e-12
```

Hemos conseguido la normalidad en solo dos variables, si fueran mas procederiamos con t test pero como no es asi, con test de Wilcoxon

- **Utilizando el comando "wilcox.test():" En cada variable transformada con respecto a la variable de resultado.**

```
p.norm <- apply(scale(datos[,1:n1]),
            2,
            function(x) wilcox.test(x~datos$Outcome)$p.value)
```

Observamos que en una primera instancia ahora todas tienen diferencias significativas, esto tenemos que corregir.

- **Realizar un ajuste de "(p valor)" para corregir el problema de la comparación múltiple.**

```
p.adj <- p.adjust(p.norm,"BH")
```

Todas siguen siendo significativas, ahora vamos a ver cuales aumentan o disminyuen respecto las otras

- **Usamos el comando "split()" dividen los datos en grupos según la variable de resultado.**
- **Calculan las medianas de cada variable para cada grupo utilizando la función "apply()" y se almacenan en el objeto "datos.median"**
- **Realizar un "data.frame" llamado "toplot" que contiene las diferencias de medianas entre los grupos y los "p.values" corregidos.**

```
datos.split <- split(datos,datos$Outcome)

datos.median <- lapply(datos.split, function(x) apply(x[,-ncol(x)],2,median))


toplot <- data.frame(medianas=Reduce("-",datos.median)
```

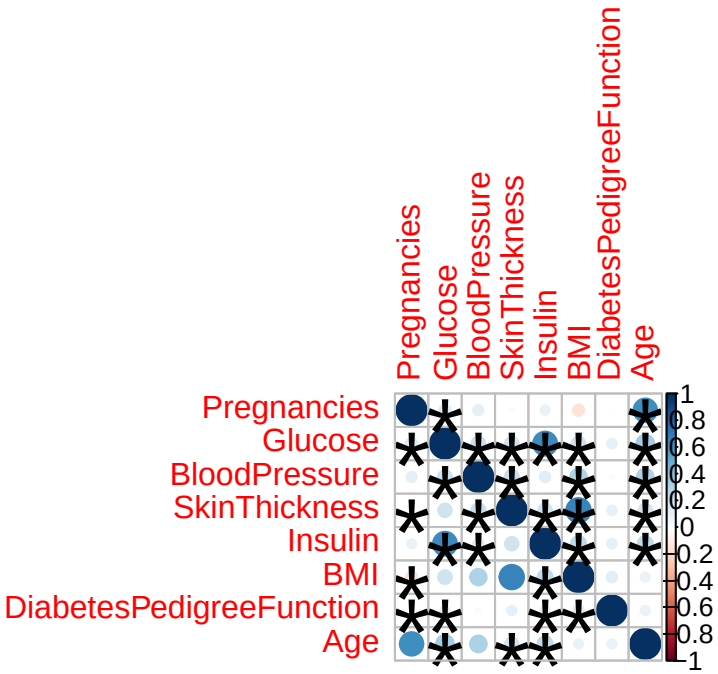
```
,p.values=p.adj)

toplot
```

```
                           medianas     p.values
Pregnancies               -0.3364722 8.957407e-05
Glucose                   -0.2957935 4.902429e-22
BloodPressure             -4.0000000 8.957407e-05
SkinThickness             -0.5484102 4.309442e-07
Insulin                   -0.4788534 3.241934e-13
BMI                       -3.3500000 2.574728e-07
DiabetesPedigreeFunction -0.2779529 8.957407e-05
Age                       -0.4005379 1.577456e-14
```

Ahora Todos los valores son significativos respecto a la obesidad

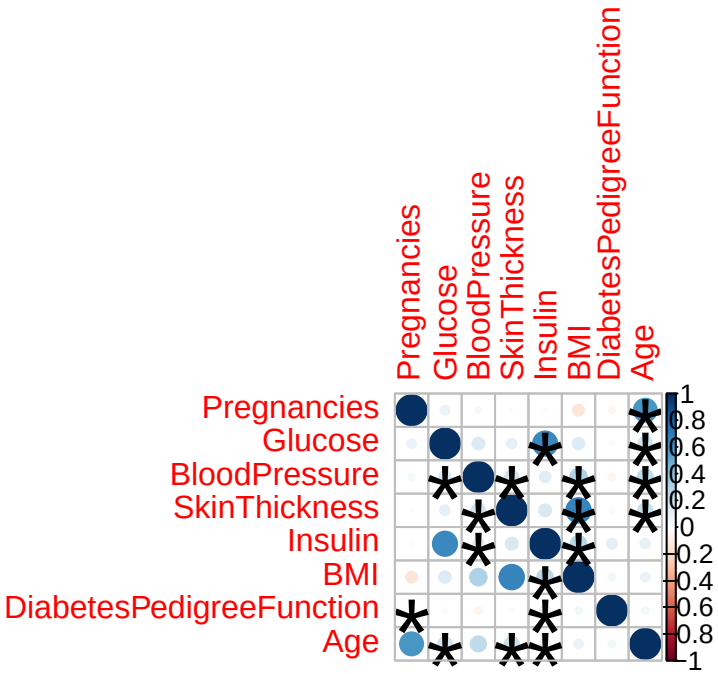
- **El comando "corr.test()" realiza una prueba de correlación en todas las variables transformadas.**
- **Los resultantes se almacenan en la funcion p.values.**
- **Corrección de "p.values" del método de Hochberg para comparación múltiple.**

```
obj.cor <- psych::corr.test(datos[,1:n1])
p.values <- obj.cor$p
p.values[upper.tri(p.values)] <- obj.cor$p.adj
p.values[lower.tri(p.values)] <- obj.cor$p.adj
diag(p.values) <- 1
corrplot::corrplot(corr = obj.cor$r,p.mat = p.values,sig.level = 0.05,insig = "label_sig")
```

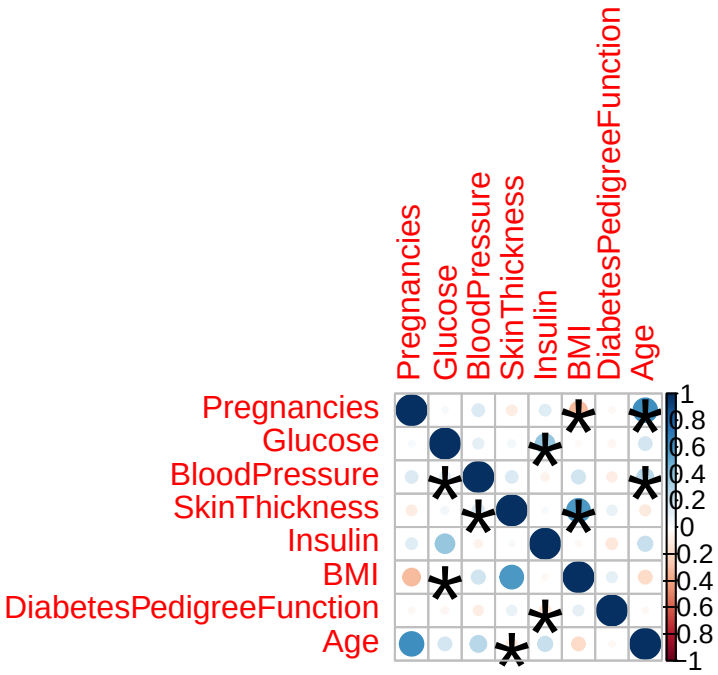


También podemos observar como cambian las relaciones segun la diabetes

```
obj.cor <- psych::corr.test(datos[datos$Outcome==0,1:n1])
p.values <- obj.cor$p
p.values[upper.tri(p.values)] <- obj.cor$p.adj
p.values[lower.tri(p.values)] <- obj.cor$p.adj
diag(p.values) <- 1
corrplot::corrplot(corr = obj.cor$r,p.mat = p.values,sig.level = 0.05,insig = "label_sig")
```




```
obj.cor <- psych::corr.test(datos[datos$Outcome==1,1:n1])
p.values <- obj.cor$p
p.values[upper.tri(p.values)] <- obj.cor$p.adj
p.values[lower.tri(p.values)] <- obj.cor$p.adj
diag(p.values) <- 1
corrplot::corrplot(corr = obj.cor$r,p.mat = p.values,sig.level = 0.05,insig = "label_sig")
```



Es decir, existen correlaciones únicas de la obesidad y no obesidad, y existen otras correlaciones que son debidas a otros factores.

Particion de datos

Partición de los datos en conjuntos de entrenamiento y prueba.

- **El comando "scale()", centra y escala cada variable para tener media cero y desviación estándar uno.**
- **El comando "levels()" cambia los niveles de la variable "Outcome" a "D" (diabetes) y "N" (no diabetes)**
- **El comando "sample()", que selecciona aleatoriamente un subconjunto de filas del "data.frame" datos para formar el conjunto de entrenamiento.**
- **El tamaño del conjunto de entrenamiento que representa el 70% de los datos.**
- **Modelo de regresión logística especifica que la variable "Outcome" es la variable de respuesta y todas las demás variables en el "data.frame" "dat.train" se utilizan como variables predictoras.**

```r
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))
levels(datos$Outcome) <- c("D","N")
train <- sample(nrow(datos),size = nrow(datos)*0.7)

dat.train <- datos[train,]
dat.test <- datos[-train,]
```

Modelado

- **El argumento "family = binomial" se utiliza para indicar que se trata de un modelo de regresión logística para datos binarios.**
- **Calcular una matriz de confusión para evaluar el rendimiento de las predicciones en comparación con los valores reales.**

```r
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))

glm.mod <- glm(Outcome ~.,data=dat.train,family = "binomial")

prediccion <- as.factor(ifelse(predict(glm.mod,dat.test,type="response")>=0.5,"N","D"))

caret::confusionMatrix(prediccion,dat.test$Outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 71 10
         N 16 21

               Accuracy : 0.7797
                 95% CI : (0.6941, 0.8507)
    No Information Rate : 0.7373
    P-Value [Acc > NIR] : 0.1737

                  Kappa : 0.4646

 Mcnemar's Test P-Value : 0.3268

            Sensitivity : 0.8161
```

```
            Specificity : 0.6774
        Pos Pred Value : 0.8765
        Neg Pred Value : 0.5676
            Prevalence : 0.7373
        Detection Rate : 0.6017
  Detection Prevalence : 0.6864
     Balanced Accuracy : 0.7468

      'Positive' Class : D
```

RIDGE

- **El comando "expand.grid" crea una cuadrícula de sintonización para el ajuste del modelo.**
- **Utilizamos método de regularización "glmnet". Lambda varía de 0 a 1 en incrementos de 0.001. Estos valores se combinan en todas las posibles combinaciones para la sintonización del modelo.**
- **Se utiliza el método de validación cruzada repetida el comando "repeatedcv" con 10 pliegues y 3 repeticiones , se entrena y evalua en 10 subconjuntos de datos diferentes, repitiendo el proceso 3 veces. La opción "classProbs = T" indica que se deben calcular las probabilidades de clase durante el entrenamiento.**

```
tuneGrid=expand.grid(
               .alpha=0,
               .lambda=seq(0, 1, by = 0.001))
trainControl <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 3,
                        # prSummary needs calculated class,
                        classProbs = T)

model <- train(Outcome ~ ., data = dat.train, method = "glmnet", trControl = trainControl,
                                        metric="Accuracy"
)

confusionMatrix(predict(model,dat.test[,-ncol(dat.test)]),dat.test$Outcome)
```

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction  D  N
         D 71 11
         N 16 20

               Accuracy : 0.7712
                 95% CI : (0.6848, 0.8435)
    No Information Rate : 0.7373
    P-Value [Acc > NIR] : 0.2345

                  Kappa : 0.4385

 Mcnemar's Test P-Value : 0.4414

            Sensitivity : 0.8161
            Specificity : 0.6452
         Pos Pred Value : 0.8659
         Neg Pred Value : 0.5556
             Prevalence : 0.7373
         Detection Rate : 0.6017
   Detection Prevalence : 0.6949
      Balanced Accuracy : 0.7306

       'Positive' Class : D
```

LASSO

- **Se aplica la misma metodologia de ridge con la única diferencia que el valor de alpha cambia a 1.**

```
tuneGrid=expand.grid(
              .alpha=1,
              .lambda=seq(0, 1, by = 0.0001))
trainControl <-


model <- train(Outcome ~ ., data = dat.train, method = "glmnet", trControl = trainControl,
                                      metric="Accuracy"
)

confusionMatrix(predict(model,dat.test[,-ncol(dat.test)]),dat.test$Outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 71 10
         N 16 21

               Accuracy : 0.7797
                 95% CI : (0.6941, 0.8507)
    No Information Rate : 0.7373
    P-Value [Acc > NIR] : 0.1737

                  Kappa : 0.4646

 Mcnemar's Test P-Value : 0.3268

            Sensitivity : 0.8161
            Specificity : 0.6774
         Pos Pred Value : 0.8765
         Neg Pred Value : 0.5676
             Prevalence : 0.7373
         Detection Rate : 0.6017
   Detection Prevalence : 0.6864
      Balanced Accuracy : 0.7468

       'Positive' Class : D
```

NAIVE BAYES

- **Se usa el comando "laplace = 0" el cual indica que no se debe aplicar el ajuste de Laplace.**
- **Este modelo tiene una accuracy del 83% y se establece cómo positivo es decir que hay diabetes.**

```r
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))
levels(datos$Outcome) <- c("D","N")
train <- sample(nrow(datos),size = nrow(datos)*0.7)

dat.train <- datos[train,]
dat.test <- datos[-train,]
mdl <- naiveBayes(Outcome ~ .,data=dat.train,laplace = 0)
```

```r
prediccion <-predict(mdl,dat.test[,-ncol(dat.test)])
confusionMatrix(prediccion,dat.test$Outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 65 13
         N 11 29

               Accuracy : 0.7966
                 95% CI : (0.7127, 0.8651)
    No Information Rate : 0.6441
    P-Value [Acc > NIR] : 0.0002321

                  Kappa : 0.5516

 Mcnemar's Test P-Value : 0.8382565

            Sensitivity : 0.8553
            Specificity : 0.6905
         Pos Pred Value : 0.8333
         Neg Pred Value : 0.7250
             Prevalence : 0.6441
         Detection Rate : 0.5508
   Detection Prevalence : 0.6610
      Balanced Accuracy : 0.7729

       'Positive' Class : D
```

- **En este código se busca el valor de lambda más cercano al mejor valor de lambda seleccionado por el modelo LASSO**

```r
lambda_use <- min(model$finalModel$lambda[model$finalModel$lambda >= model$bestTune$lambda
position <- which(model$finalModel$lambda == lambda_use)
featsele <- data.frame(coef(model$finalModel)[, position])
```

- **La función rownames() para obtener los nombres de las variables seleccionadas por el modelo LASSO**

```
rownames(featsele)[featsele$coef.model.finalModel....position.!=0]
```

```
[1] "(Intercept)"              "Pregnancies"
[3] "Glucose"                  "SkinThickness"
[5] "BMI"                      "DiabetesPedigreeFunction"
[7] "Age"
```

- **El comando "naiveBayes()" se usa para ajustar el modelo Naive Bayes utilizal as variables seleccionadas.**
- **El comando "predict()" para realizar las predicciones en el conjunto de prueba "dat.test\[, -ncol(dat.test)\]", excluyendo la columna de la variable objetivo.**

```
mdl.sel <-naiveBayes(Outcome ~ Insulin+Glucose+DiabetesPedigreeFunction+Age,data = dat.tra

prediccion <- predict(mdl.sel,dat.test[,-ncol(dat.test)])

confusionMatrix(prediccion,dat.test$Outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 65 13
         N 11 29

               Accuracy : 0.7966
                 95% CI : (0.7127, 0.8651)
    No Information Rate : 0.6441
    P-Value [Acc > NIR] : 0.0002321

                  Kappa : 0.5516

 Mcnemar's Test P-Value : 0.8382565

            Sensitivity : 0.8553
            Specificity : 0.6905
         Pos Pred Value : 0.8333
         Neg Pred Value : 0.7250
             Prevalence : 0.6441
```

```
      Detection Rate : 0.5508
Detection Prevalence : 0.6610
   Balanced Accuracy : 0.7729

    'Positive' Class : D
```

- **El método utilizado es "knn" y se especifica "preProcess = c("center", "scale")" para centrar y escalar las variables predictoras**

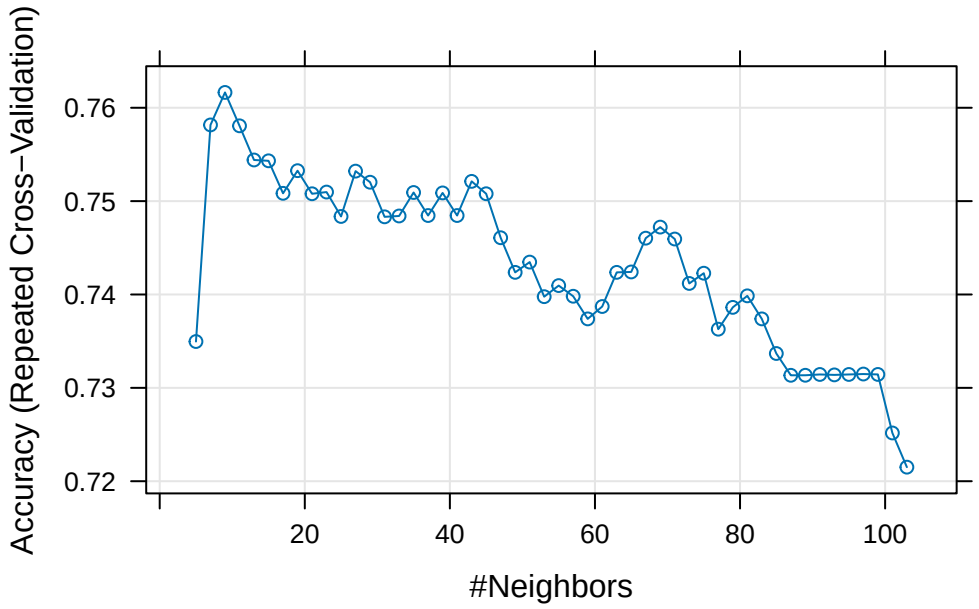
```
library(ISLR)
library(caret)
set.seed(400)
ctrl <- trainControl(method="repeatedcv",repeats = 3) #,classProbs=TRUE,summaryFunction =
knnFit <- train(Outcome ~ ., data = dat.train, method = "knn", trControl = ctrl, preProces

#Output of kNN fit
knnFit
```

```
k-Nearest Neighbors

274 samples
  8 predictor
  2 classes: 'D', 'N'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 246, 248, 246, 246, 248, 246, ...
Resampling results across tuning parameters:

  k    Accuracy   Kappa
    5  0.7349613  0.3286463
    7  0.7581604  0.3812300
    9  0.7616402  0.3795962
   11  0.7580654  0.3617266
   13  0.7544058  0.3516751
   15  0.7543176  0.3470807
   17  0.7508377  0.3372968
   19  0.7532628  0.3446640
   21  0.7507869  0.3359935
   23  0.7509700  0.3286132
   25  0.7483584  0.3242062
```

```
  27  0.7532119  0.3285274
  29  0.7520248  0.3318467
  31  0.7483177  0.3270561
  33  0.7484093  0.3287037
  35  0.7509259  0.3393297
  37  0.7484568  0.3281433
  39  0.7508784  0.3316511
  41  0.7484568  0.3243866
  43  0.7521164  0.3321338
  45  0.7507903  0.3274954
  47  0.7460792  0.3159212
  49  0.7423721  0.3010017
  51  0.7434710  0.3042315
  53  0.7397673  0.2939945
  55  0.7409544  0.2927135
  57  0.7398114  0.2895922
  59  0.7373864  0.2785776
  61  0.7387125  0.2847708
  63  0.7423721  0.2935447
  65  0.7424128  0.2961597
  67  0.7460317  0.3007357
  69  0.7472188  0.3033893
  71  0.7459402  0.3017267
  73  0.7411817  0.2841383
  75  0.7422806  0.2842496
  77  0.7362841  0.2660344
  79  0.7386209  0.2712211
  81  0.7398555  0.2736582
  83  0.7373864  0.2643776
  85  0.7336793  0.2476887
  87  0.7313458  0.2347981
  89  0.7313492  0.2349988
  91  0.7314340  0.2296727
  93  0.7313899  0.2249436
  95  0.7314340  0.2235843
  97  0.7314815  0.2184058
  99  0.7314374  0.2180500
 101  0.7251696  0.1971713
 103  0.7215066  0.1815100

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 9.
```

- **Se usa el comando "plot(knnFit)" para graficar el modelo.**

```
plot(knnFit)
```



- **El modelo knn entrenado se usa en la predicción y la matriz de confusión se usa para evaluar los parámetros de rendimiento y la precisión del modelo.**

```
knnPredict <- predict(knnFit,newdata = dat.test[,-ncol(dat.test)] )
#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(knnPredict, dat.test$Outcome )
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 69 18
         N  7 24

               Accuracy : 0.7881
                 95% CI : (0.7033, 0.858)
    No Information Rate : 0.6441
    P-Value [Acc > NIR] : 0.0005023
```

```
                 Kappa : 0.5092

Mcnemar's Test P-Value : 0.0455003

            Sensitivity : 0.9079
            Specificity : 0.5714
         Pos Pred Value : 0.7931
         Neg Pred Value : 0.7742
             Prevalence : 0.6441
         Detection Rate : 0.5847
   Detection Prevalence : 0.7373
      Balanced Accuracy : 0.7397

       'Positive' Class : D
```

- **Se ajusta del modelo PLS-DA utilizando la función "train()"**
- **Luego de ajustar el modelo PLS-DA, se realiza las predicciones en el conjunto de prueba utilizando el comando "predict()"**
- **El comando "confusionMatrix()" sirve para calcular la matriz de confusión y obtener medidas de evaluación.**

```r
library(caret)
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <-as.factor(datos$Outcome)
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))
levels(datos$Outcome) <- c("D","N")
train <- sample(nrow(datos),size = nrow(datos)*0.7)

dat.train <- datos[train,]
dat.test <- datos[-train,]
set.seed(1001)
ctrl<-trainControl(method="repeatedcv",number=10,classProbs = TRUE,summaryFunction = twoCl
plsda<-train(x=dat.train[,-ncol(datos)], # spectral data
             y=dat.train$Outcome, # factor vector
             method="pls", # pls-da algorithm
             tuneLength=10, # number of components
             trControl=ctrl, # ctrl contained cross-validation option
             preProc=c("center","scale"), # the data are centered and scaled
             metric="ROC") # metric is ROC for 2 classes
```

```
plsda
```

```
Partial Least Squares

537 samples
  8 predictor
  2 classes: 'D', 'N'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 483, 484, 483, 483, 483, 483, ...
Resampling results across tuning parameters:

  ncomp  ROC        Sens       Spec
  1      0.8183485  0.8468067  0.5657895
  2      0.8348713  0.8667227  0.6181579
  3      0.8346068  0.8814286  0.6023684
  4      0.8342848  0.8756303  0.6076316
  5      0.8338425  0.8784874  0.6023684
  6      0.8336922  0.8784874  0.6023684
  7      0.8336922  0.8784874  0.6023684

ROC was used to select the optimal model using the largest value.
The final value used for the model was ncomp = 2.
```

```
prediccion <- predict(plsda,newdata = dat.test[,-ncol(datos)])

confusionMatrix(prediccion,dat.test$Outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   D   N
         D 135  40
         N  19  37

               Accuracy : 0.7446
                 95% CI : (0.6833, 0.7995)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 0.006419
```

```
                 Kappa : 0.3833

 Mcnemar's Test P-Value : 0.009220

            Sensitivity : 0.8766
            Specificity : 0.4805
         Pos Pred Value : 0.7714
         Neg Pred Value : 0.6607
             Prevalence : 0.6667
         Detection Rate : 0.5844
   Detection Prevalence : 0.7576
      Balanced Accuracy : 0.6786

       'Positive' Class : D
```

Si tuneamos lambda

- **El rendimiento general del modelo Naive Bayes en el conjunto de prueba se mostrará en el resultado impreso.**

```r
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <-as.factor(datos$Outcome)
levels(datos$Outcome) <- c("D","N")
train <- sample(nrow(datos),size = nrow(datos)*0.7)

dat.train <- datos[train,]
dat.test <- datos[-train,]
lambda <- seq(0,50,0.1)

  modelo <- naiveBayes(dat.train[,-ncol(datos)],dat.train$Outcome)

  predicciones <- predict(modelo,dat.test[,-ncol(datos)])

confusionMatrix(predicciones,dat.test$Outcome)$overall[1]
```

```
 Accuracy
0.7705628
```

- **El código carga y prepara los datos de diabetes, divide los datos en conjuntos de prueba y entrenamiento, ajusta un modelo Naive Bayes y produce predic-**

ciones usando el modelo ajustado. Después de eso, la matriz de confusión se usa para medir la precisión general del modelo.

```
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <-as.factor(datos$Outcome)
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))
levels(datos$Outcome) <- c("D","N")
train <- sample(nrow(datos),size = nrow(datos)*0.7)

dat.train <- datos[train,]
dat.test <- datos[-train,]
library(caret)
set.seed(1001)
ctrl<-trainControl(method="repeatedcv",number=10,classProbs = TRUE,summaryFunction = twoCl
plsda<-train(x=dat.train[,c(2,5,7,8)], # spectral data
             y=dat.train$Outcome, # factor vector
             method="pls", # pls-da algorithm
             tuneLength=10, # number of components
             trControl=ctrl, # ctrl contained cross-validation option
             preProc=c("center","scale"), # the data are centered and scaled
             metric="ROC") # metric is ROC for 2 classes

prediccion <- predict(plsda,dat.test[,c(2,5,7,8)])
confusionMatrix(prediccion,dat.test$Outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   D   N
         D 136  42
         N   9  44

               Accuracy : 0.7792
                 95% CI : (0.7201, 0.831)
    No Information Rate : 0.6277
    P-Value [Acc > NIR] : 5.532e-07

                  Kappa : 0.4876

 Mcnemar's Test P-Value : 7.433e-06

            Sensitivity : 0.9379
```

```
           Specificity : 0.5116
        Pos Pred Value : 0.7640
        Neg Pred Value : 0.8302
            Prevalence : 0.6277
        Detection Rate : 0.5887
  Detection Prevalence : 0.7706
     Balanced Accuracy : 0.7248

      'Positive' Class : D
```

Finalmente podríamos hacer un análisis de la varianza multivariante

- El comando "adonis2" realiza un análisis de varianza multivariante utilizando la disimilitud indicada por la distancia euclidiana.

```
library(vegan)
```

```
Loading required package: permute
```

```
This is vegan 2.6-4
```

```

Attaching package: 'vegan'
```

```
The following object is masked from 'package:caret':

    tolerance
```

```
adonis2(datos[,-ncol(datos)] ~datos$Outcome,method = "euclidean")
```

```
Permutation test for adonis under reduced model
Terms added sequentially (first to last)
Permutation: free
Number of permutations: 999

adonis2(formula = datos[, -ncol(datos)] ~ datos$Outcome, method = "euclidean")
               Df SumOfSqs      R2      F Pr(>F)
datos$Outcome   1    357.8 0.05831 47.434  0.001 ***
```

```
Residual      766   5778.2 0.94169
Total         767   6136.0 1.00000
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Es decir, como conlusión aunque las variables no pueden detectar la diabetes, siendo variables independientes, si por otro lado las consideramos dependientes de la diabetes.

Es decir, la diabetes es una condición en la que influye en los parámetros, mientras que es menos probable que la diabetes sea la causa de estas alteraciones, con una mejor precisón del 77 por ciento.

Es decir, por un lado tenemos las variables que nos explican solo un 77 porciento de la diabetes, mientras que la condición en sí nos separa más entre la media global.

Se podría investigar más esto. Por ejemplo, se podría hacer una correlación parcial, dada la diabetes, e identificar aquellas variables especificamente relacionadas con esta.

CURVA ROC

Se utilizará la biblioteca pROC para ello instalarla con el siguiente código: "(install.packages("pROC"))"

La curva ROC representa:

- **La relación entre la tasa de verdaderos positivos**
- **La tasa de falsos positivos (1-especificidad) a medida que se va variando el umbral de clasificación.**

```
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```r
# Regresión Logística
glm.mod <- glm(Outcome ~ ., data = dat.train, family = "binomial")

# Probabilidades
pred.prob <- predict(glm.mod, dat.test, type = "response")

# Crear
roc_obj <- roc(dat.test$Outcome, pred.prob)
```

```
Setting levels: control = D, case = N
```

```
Setting direction: controls < cases
```

```r
# Graficar
plot(roc_obj, main = "Curva ROC", xlab = "Tasa de Falsos Positivos", ylab = "Tasa de Verda
```