Predicción de la diabetes

Katherine Criollo, Christian Guanoquiza, Esteban Narea

Intro

Este sería un ejemplo de examen El siguiente conjunto de datos, consuste 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.

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.

Cargamos librerias

```
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
```

Cargamos los datos

```
datos <- read.csv("./datos/diabetes.csv")#en esta sección s ecargan los datos del archivo
head(datos)#Este comando sirve para mostrar la primera fila de la columna de datos cargado
```

```
  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

```
str(datos)#proporciona información sobre la estructura del objeto, mostrando el tipo de da
```

```
'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

```
datos$Outcome  <- as.factor(datos$Outcome)# se llama a la columna OUTCOME la cual está con
```

Análisis estadístico preliminar

Se realiza este análisis ya que aún no se tiene definido la idea de la relación entre los datos y las variables.

```
dim(datos)#la función "dim! tiene el objetivo de devolver un vector con dos elementos, el
```

```
[1] 768   9
```

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

Histogramas

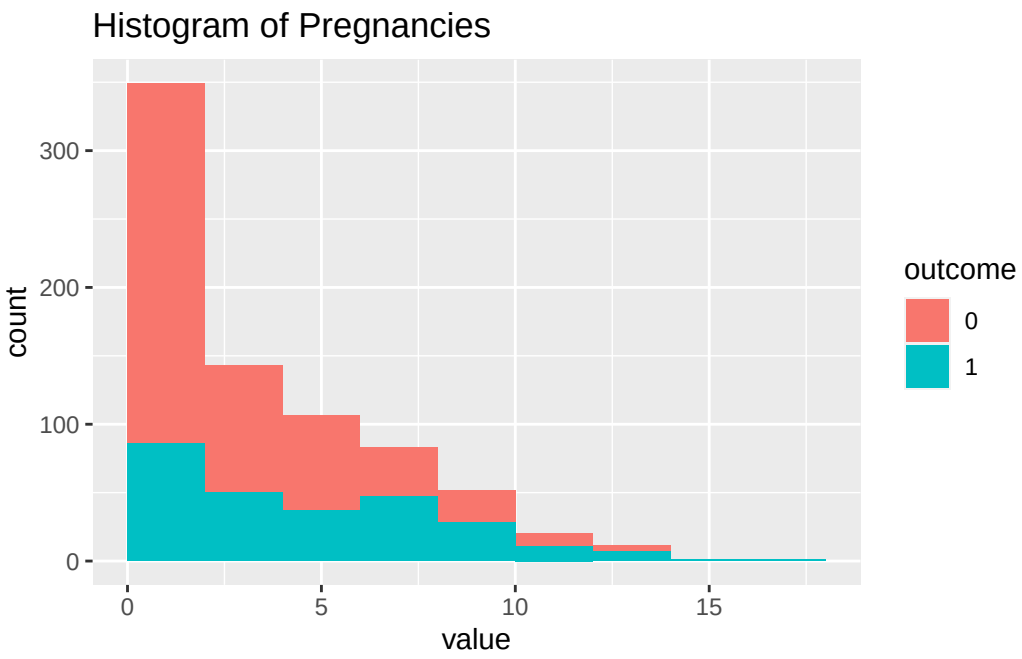
Histograma de los datos cargados.

```
l.plots <- vector("list",length = ncol(datos)-1)#se crea el vector con el nombre L.plots e
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
}
#es un vector con una longitud de datos igual al númeor de columnas menos unos, es decir a
#El bucle for revisa los datos y los recorre de j desde 1 hasta la cantidad máxima de n1.
```

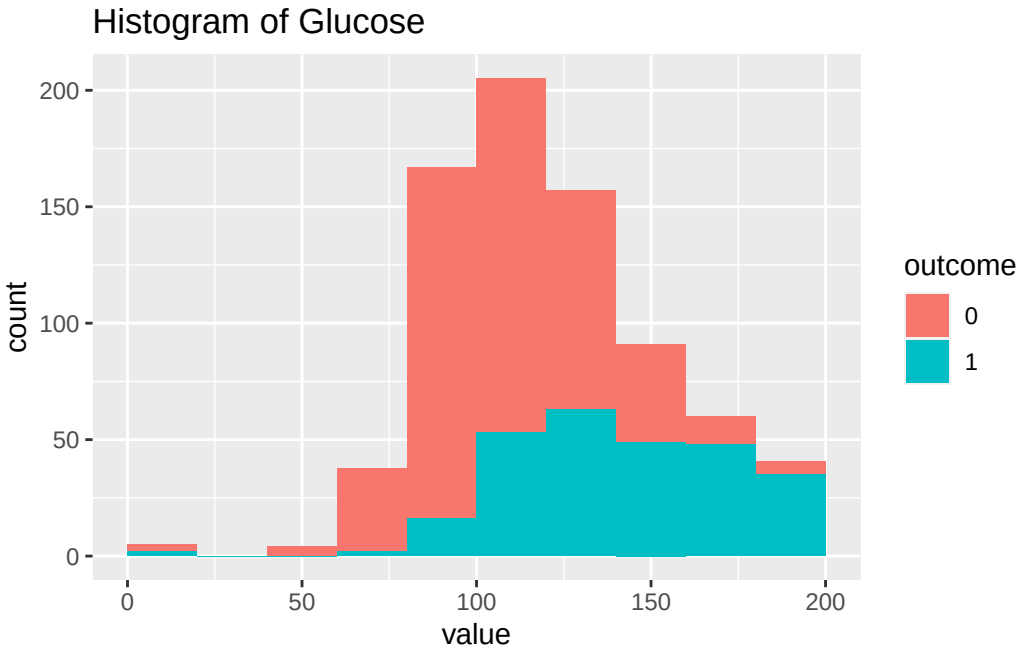
Mostrar los histogramas. Estos histogramas se representan para cada variable que en total son 8.

```
l.plots #Muestra los histogramas para cada variable con el fin de mostrar la distribución
```

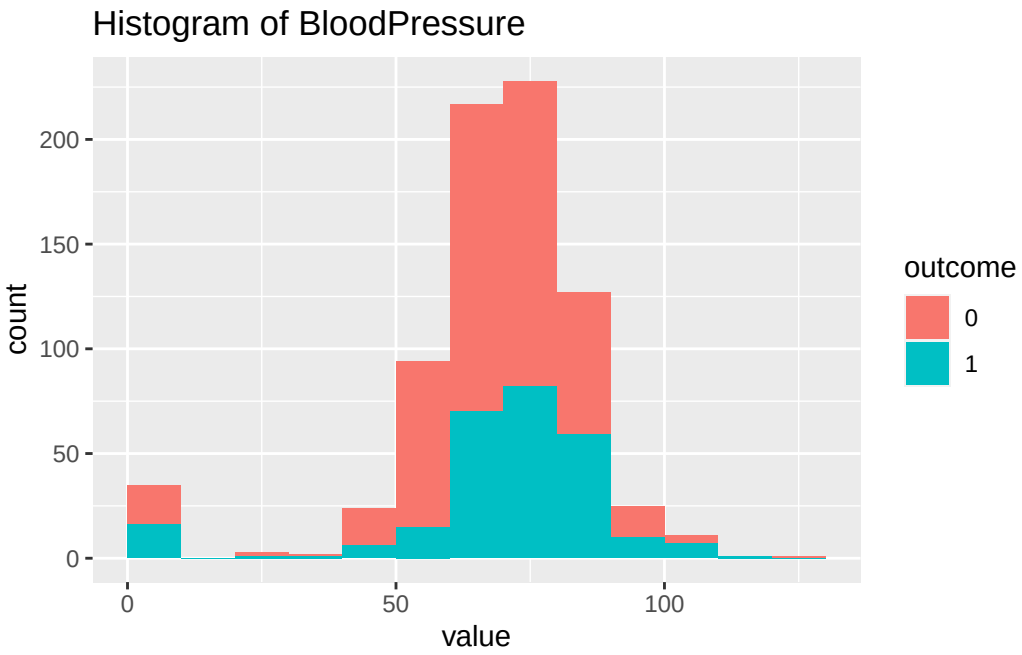
```
[[1]]
```


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

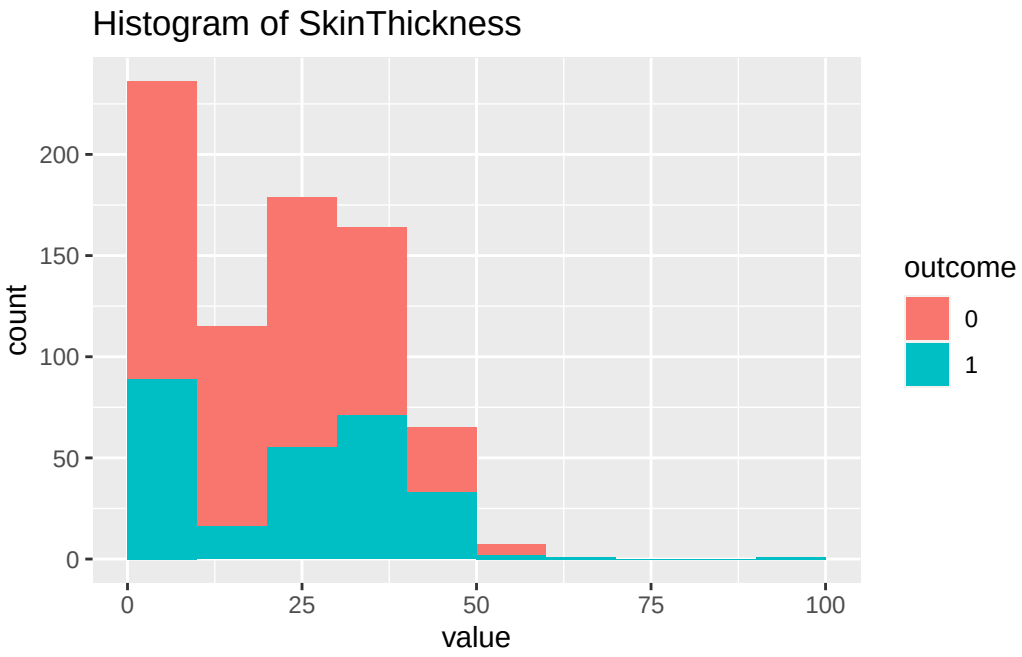
```
[[2]]
```


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

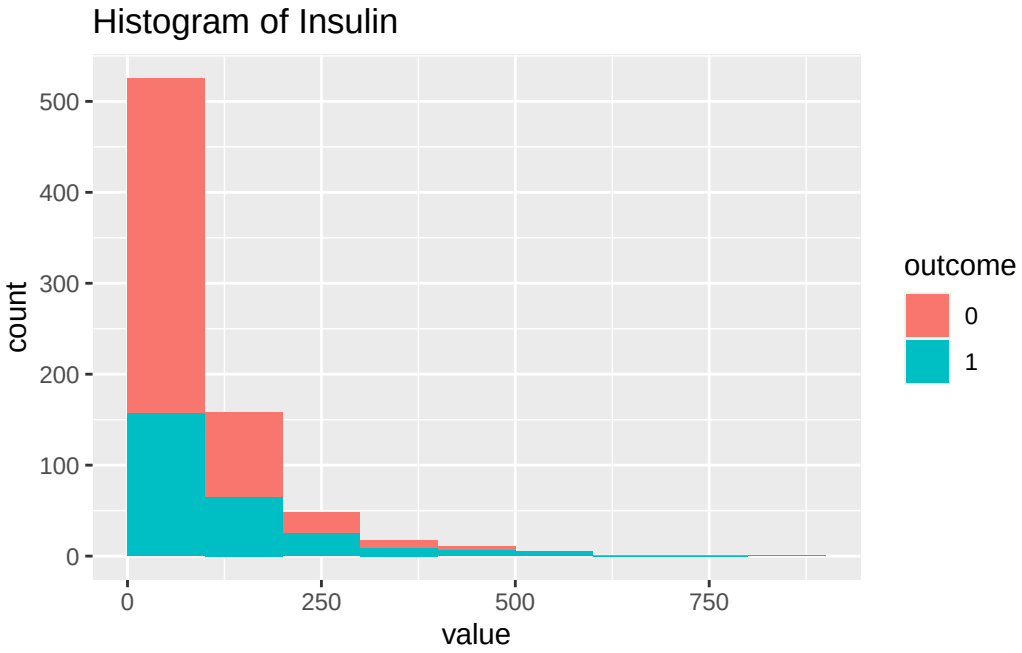
```
[[3]]
```


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

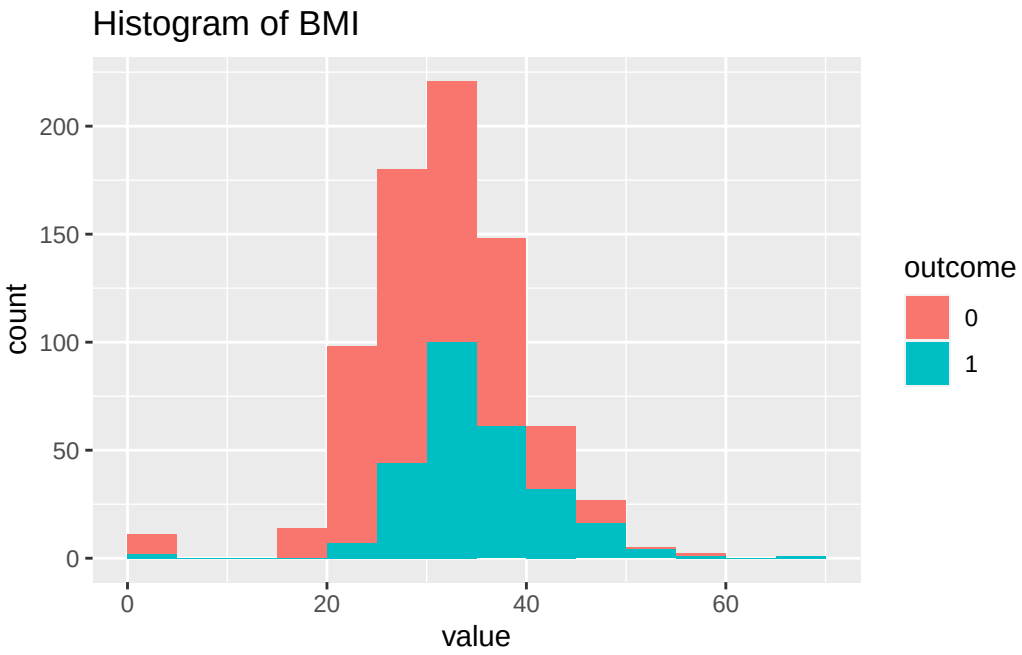
```
[[4]]
```


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

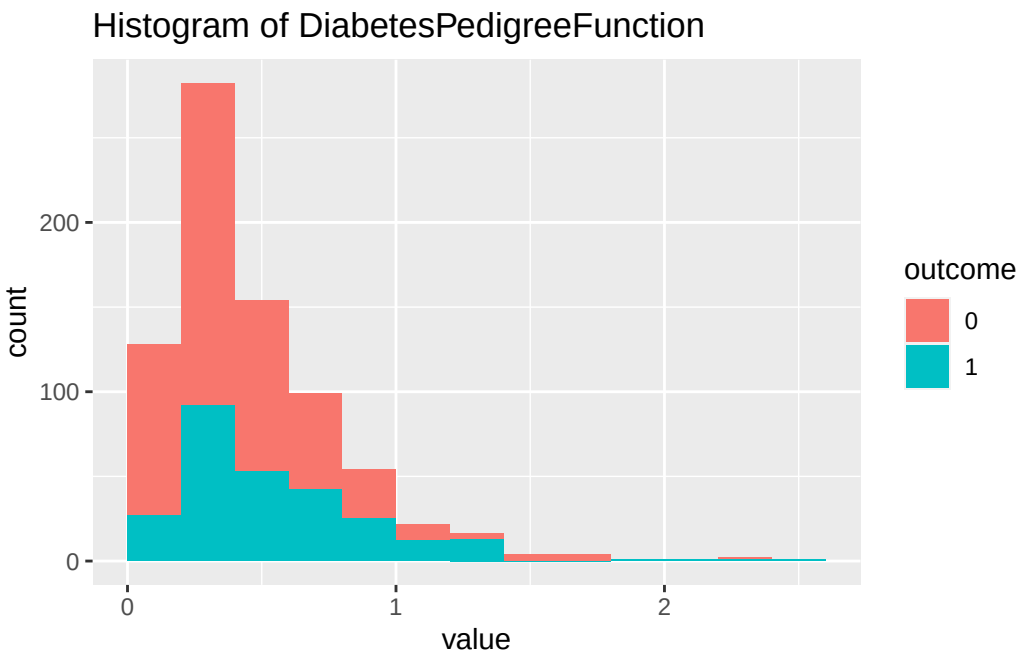
```
[[5]]
```


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

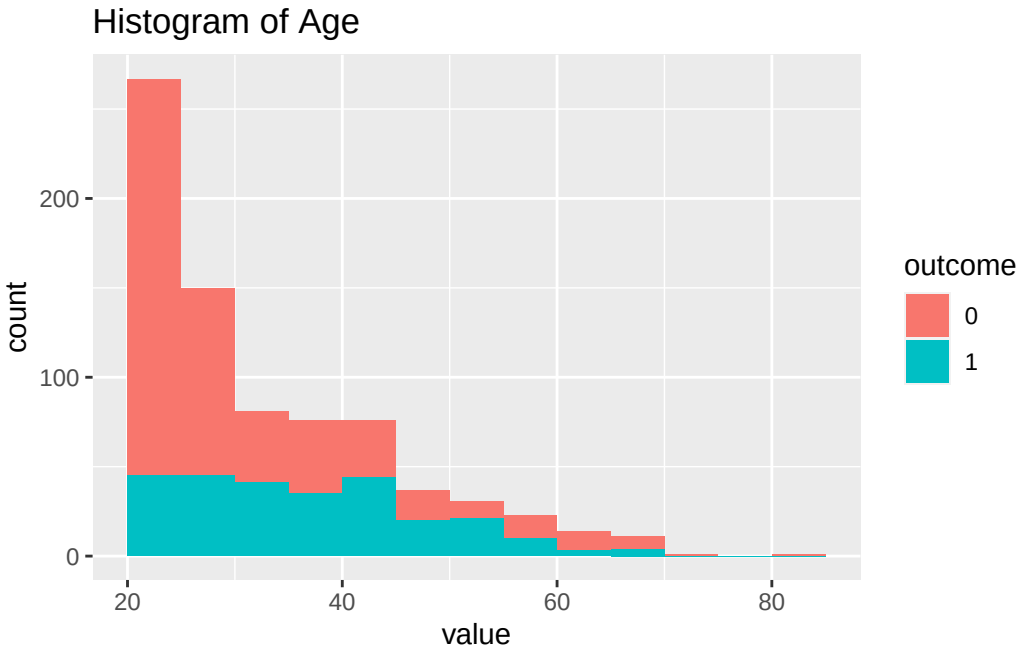
[[6]]


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...

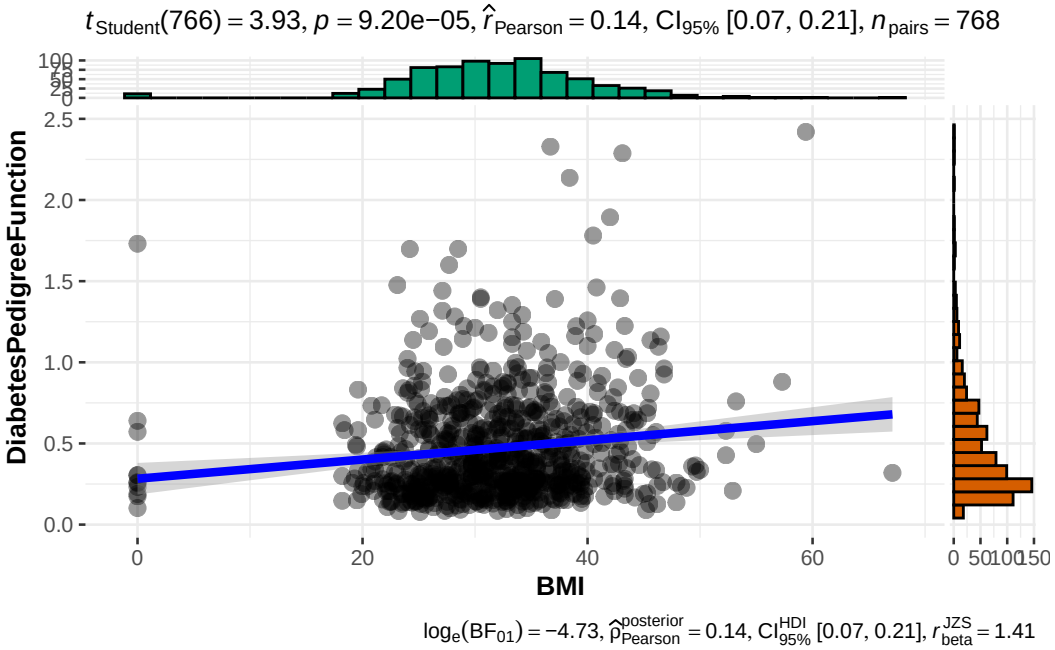
Gráfico de dispersión

Este gráfico sirve para mostrar los valores de dos variables para el conjunto de datos. Este tiene el objetivo de visualizan la relación entre dos variables numéricas, de forma que una variable se muestra en el eje x y la otra, en el eje y.

```
ggscatterstats(datos,BMI,DiabetesPedigreeFunction)# Esta rfunción nos genera un gráfico de
```

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

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

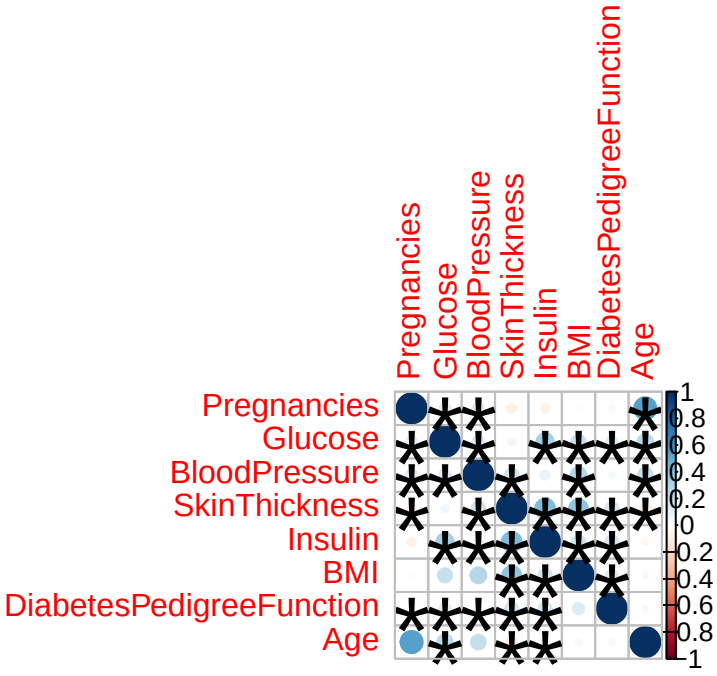


Sin embargo, esto puede ser un proceso tedioso… imaginad hacer 16 gráficas ! podemos condersarlo todo

Simplificación:

Matriz de correlación.

```
obj.cor <- psych::corr.test(datos[,1:n1])# se utiliza un teste de correlación con el finde
p.values <- obj.cor$p# Se estraen los valores del objeto "obj.cor" los cuales representan
p.values[upper.tri(p.values)] <- obj.cor$p.adj #se crea una matriz booleana que selecciona
p.values[lower.tri(p.values)] <- obj.cor$p.adj# se realiza un proceso similar al anterior
diag(p.values) <- 1# se establece en 1 la diagonal principal de la matriz con los valores
corrplot::corrplot(corr = obj.cor$r,p.mat = p.values,sig.level = 0.05,insig = "label_sig")
```



La matriz de correlación **muestra los valores de correlación, que miden el grado de relación lineal entre cada par de variables**. Los valores de correlación se pueden ubicar entre -1 y +1. Si las dos variables tienden a aumentar o disminuir al mismo tiempo, el valor de correlación es positivo.

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

Prueba de normalidad de Shapiro- Wilk

Este tipo de prueba se usa para contrastar la normalidad de un conjunto de datos.

Es aplicable cuando se analizan muestras compuestas por menos de 50 elementos (muestras pequeñas).

```
p.norm <- apply(apply(datos[,1:n1],# se ajusta un modelo de regresión lineal para cada col
            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
```

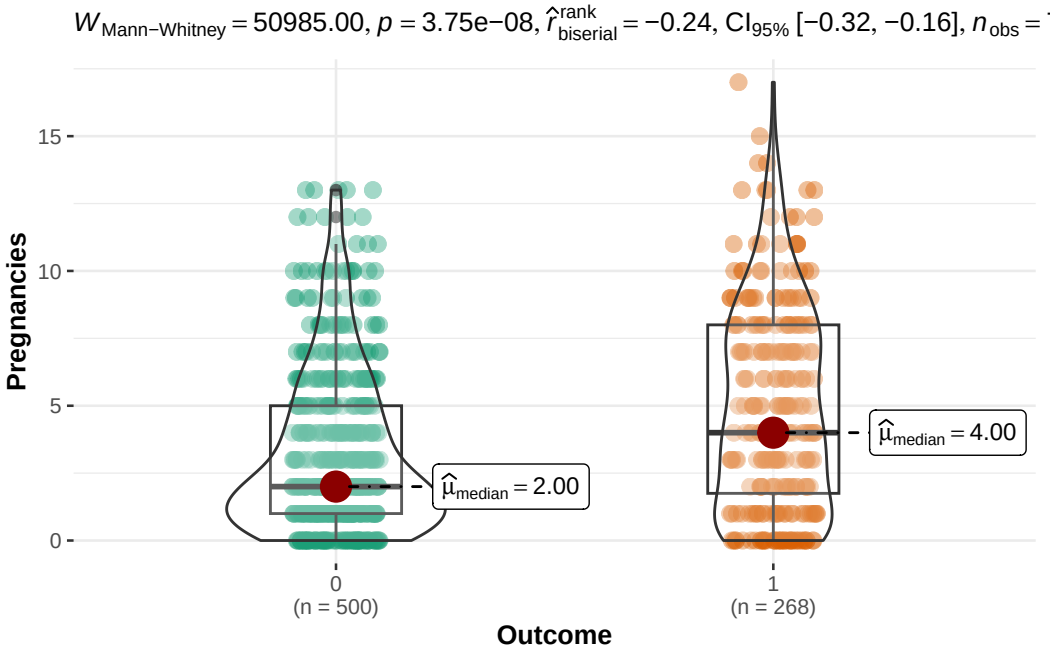
```
$SkinThickness

    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
```

```
#Se aplica la prueba de normalidad de Shapiro- Wilk a los residuos de cada varible.
#Todos los resultados se documentan en la variable: p.norm
```

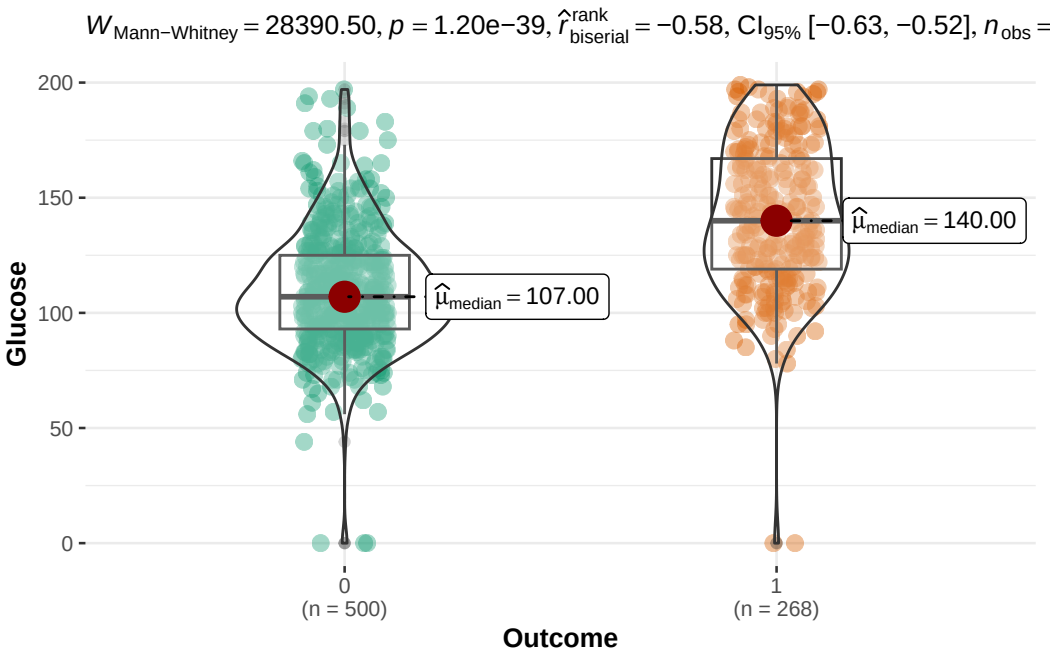
Todas las variables son no normales, tal como vemos en los histogramas.

type = "nonparametric" = Indica que se debe utilizar un enfoque no paramétrico para el análisis.

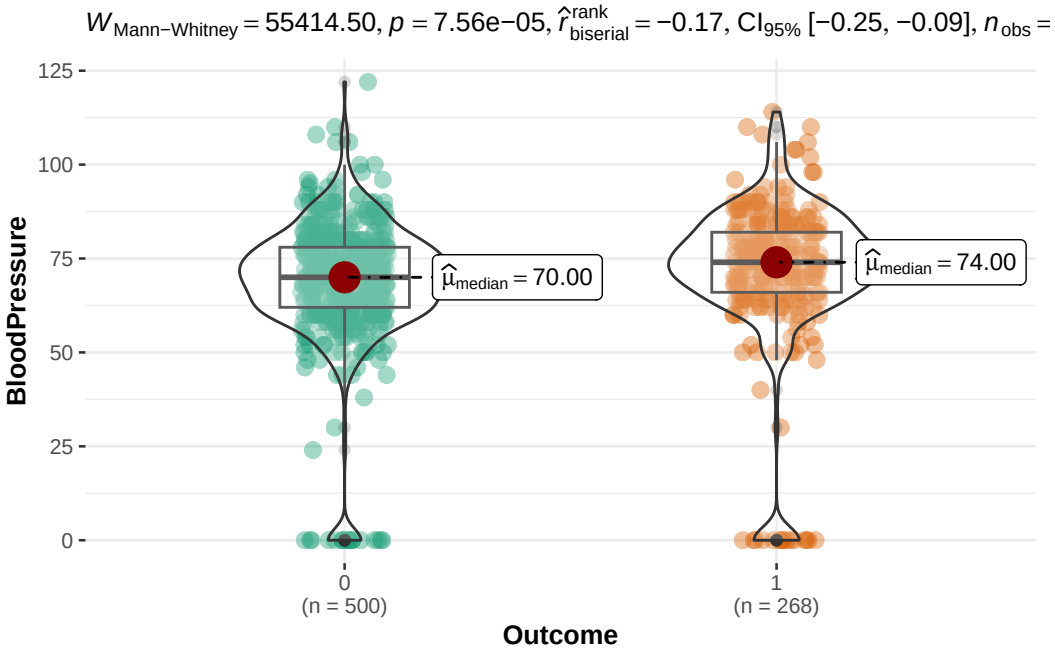
```
ggbetweenstats(datos,Outcome,Pregnancies,type = "nonparametric") #realiza un análisis de d
```



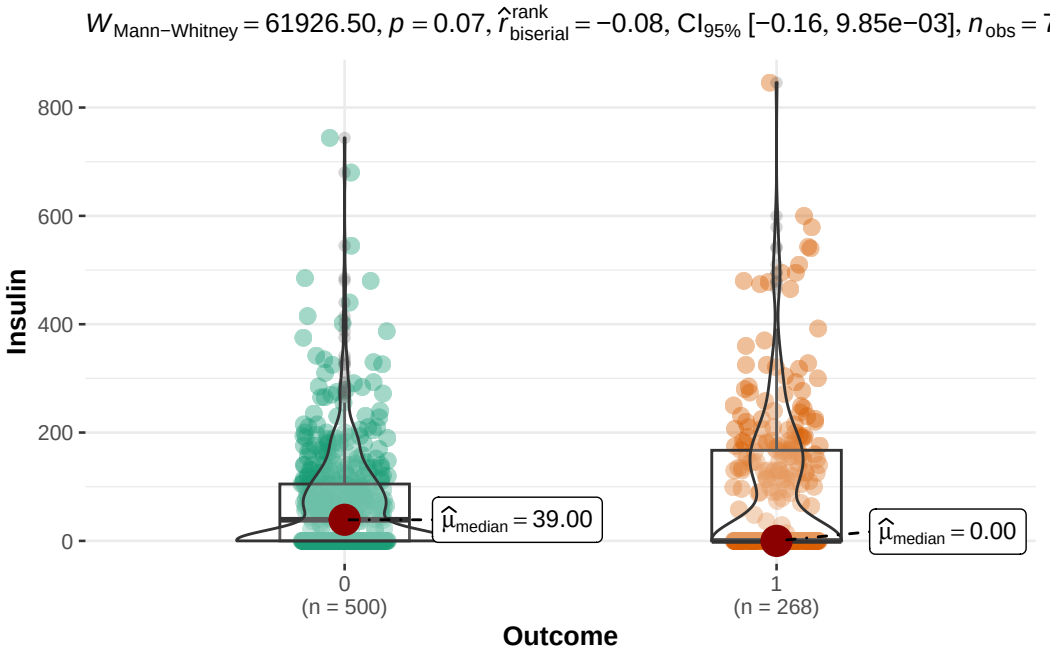

```
ggbetweenstats(datos,Outcome,Glucose,type = "nonparametric") # #realiza un análisis de dif
```



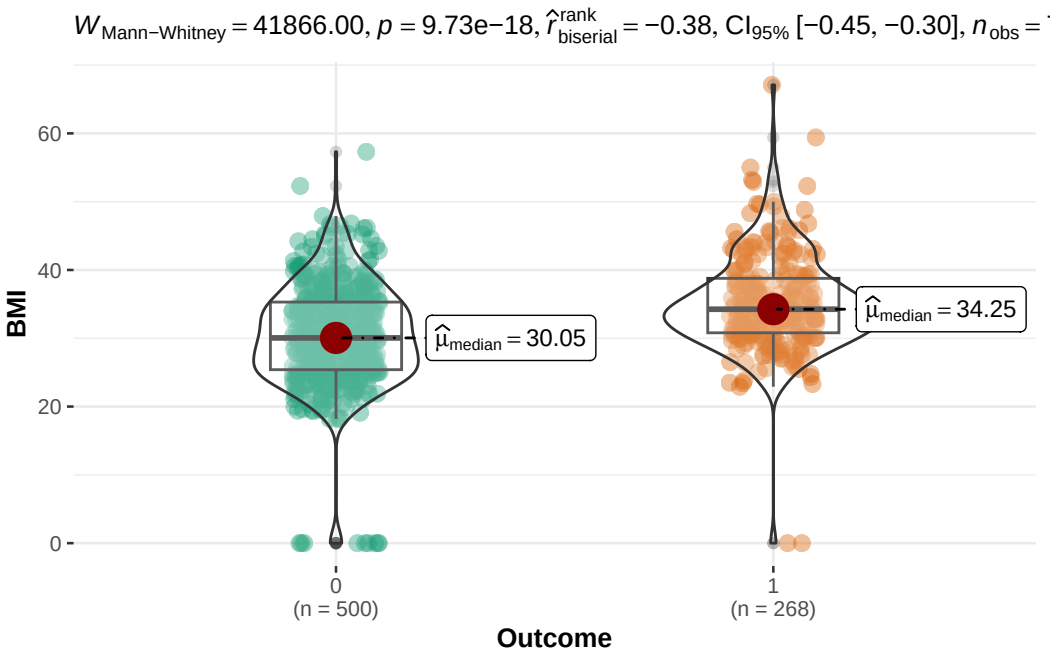

```
ggbetweenstats(datos,Outcome,BloodPressure,type = "nonparametric")# #realiza un análisis d
```



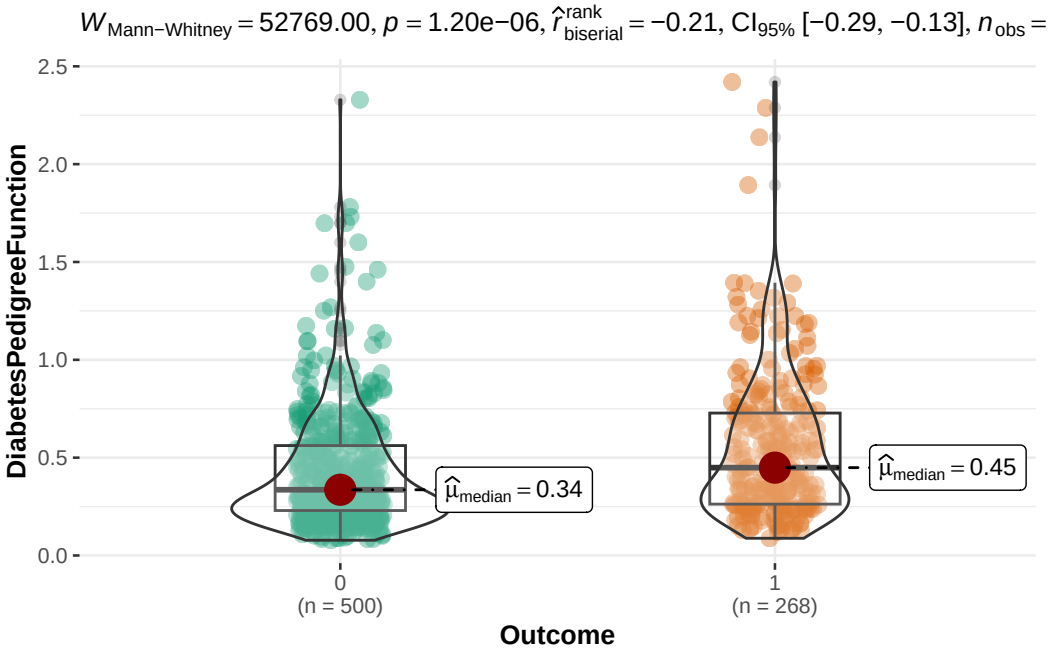
```
ggbetweenstats(datos,Outcome,Insulin,type = "nonparametric")# #realiza un análisis de dife
```



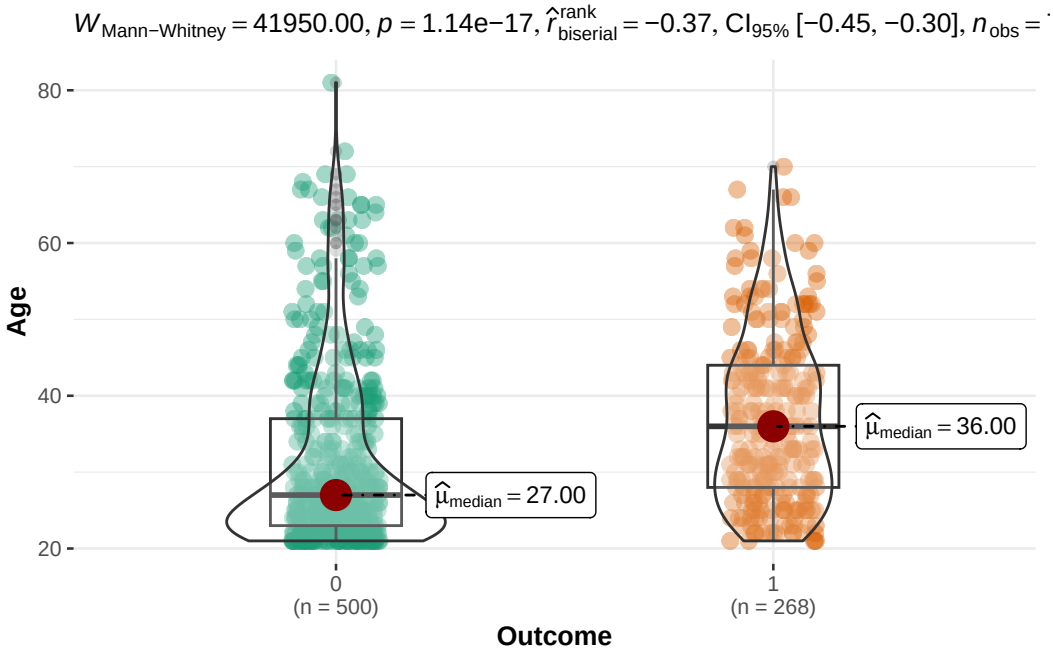

```
ggbetweenstats(datos,Outcome,BMI,type = "nonparametric")# #realiza un análisis de diferenc
```




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



```
ggbetweenstats(datos,Outcome,Age,type = "nonparametric")# #realiza un análisis de diferenc
```



PCA

Análisis de componentes principales sirve para reducir el número de variables de forma que pasemos a tener el mínimo número de nuevas variables y que representen a todas las antiguas variables de la forma más representativa posible.

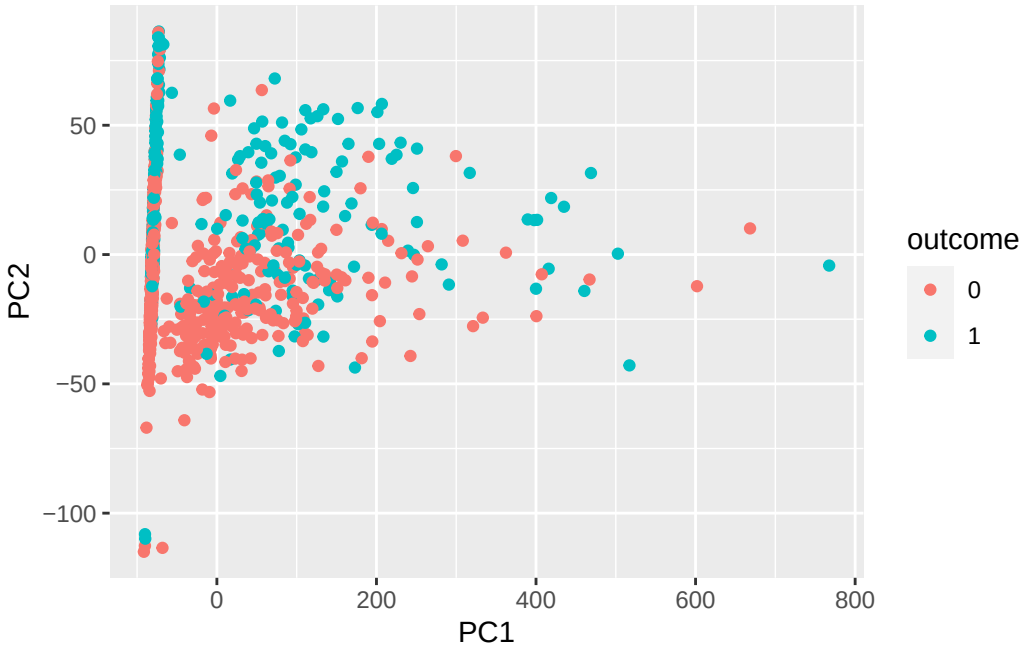
El objetivo es crear un gráfico de dispersión de los componentes principales para los valores PC1 y PC2 utilizando la función ggplot.

```
summary(datos)#resumen estadístico de los 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
#se realiza un análisis de componentes principales
plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)#combinación de las variables categóricas
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()# gráfico de dispersión de los comp
```



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...

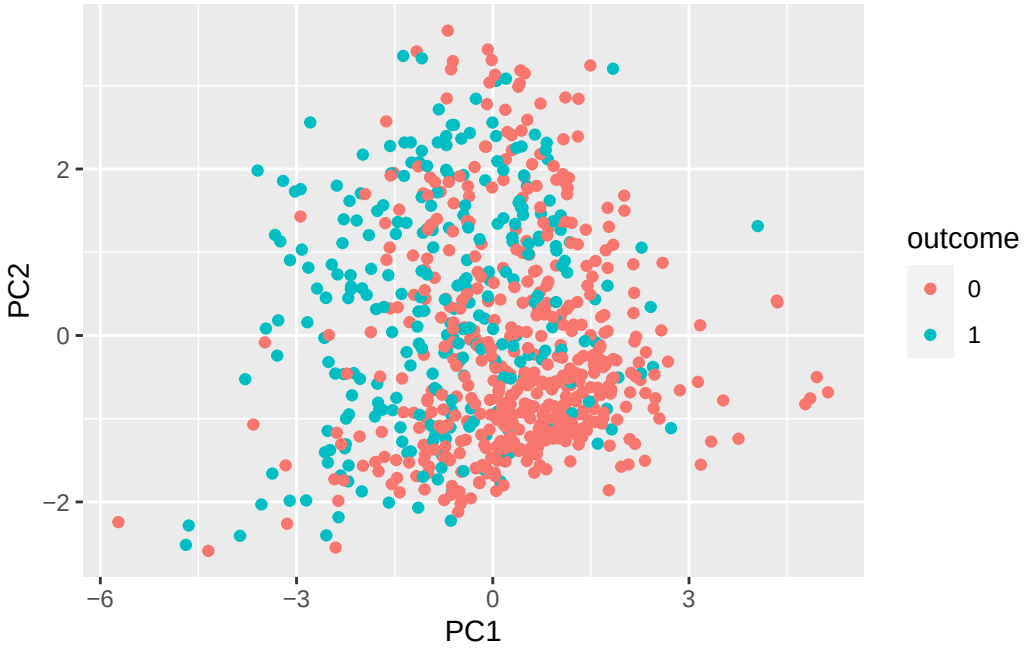
Gráfico de dispersión

Los gráficos de dispersión se usan **para averiguar la intensidad de la relación entre dos variables numéricas**. El eje X representa la variable independiente, mientras que el eje Y representa la variable dependiente.

```
summary(datos)#resumen estadístico de los datos
```

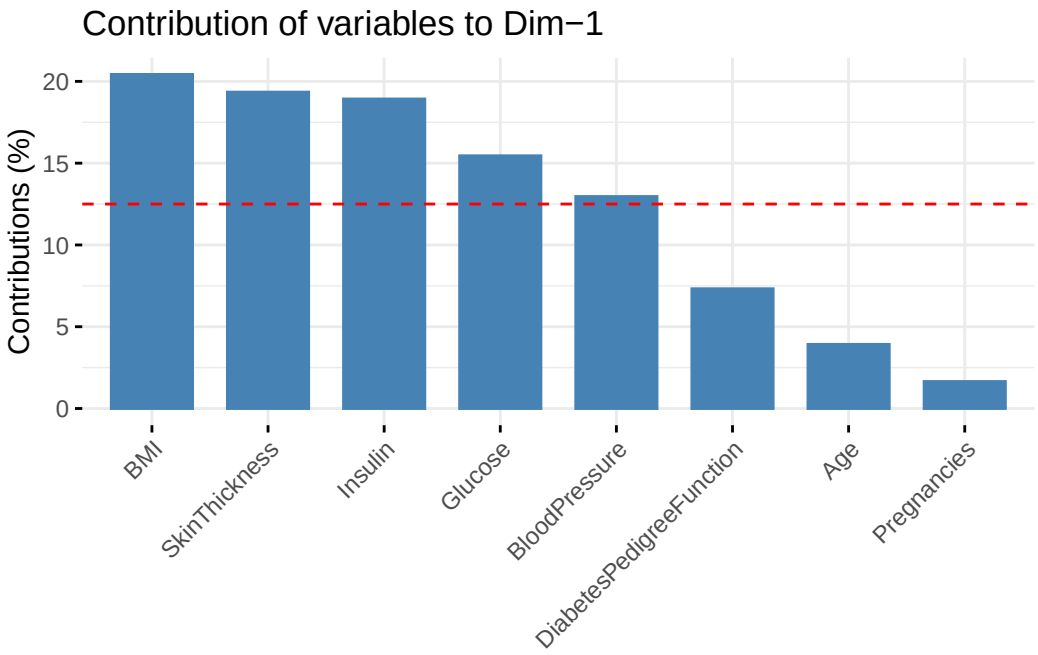
```
 Pregnancies        Glucose      BloodPressure    SkinThickness
Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
Mean   : 3.845   Mean   :120.9   Mean   : 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
# análisis de componente principales.
plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()# se utiliza la librería ggplot2 pa
```



Contribuciones de las variables originales.

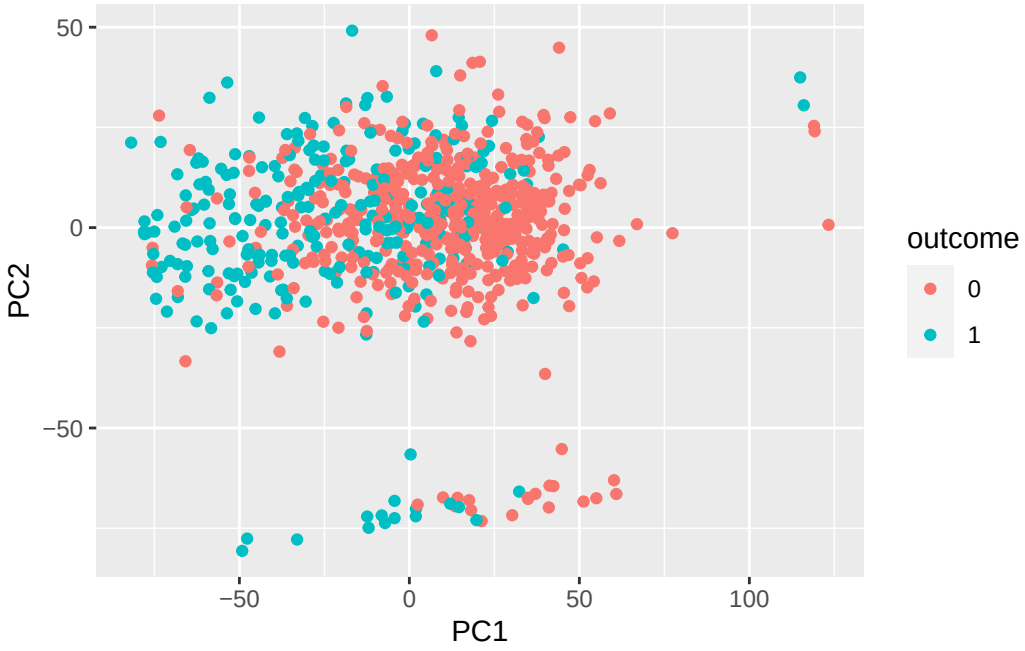
```
factoextra::fviz_contrib(pcx,"var")# aquí se visualiza las contribuciones de las variables
```



Al parecer es la insulina la que está dando problemas

Se realiza un análisis de componentes principales excluyendo a los datos de insulina.

```
## indices a quitar
w <- c(grep("insulin",ignore.case = T,colnames(datos)),ncol(datos))# se crea el vector W e
pcx <- prcomp(datos[,-w],scale. = F) ## escalamos por la variablidad de los datos
# nuevo análisis de componente principales, peroe sta vez se exluye a los datos de insulin
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 aplica la transformada logarítmica a la variable a Insulina.

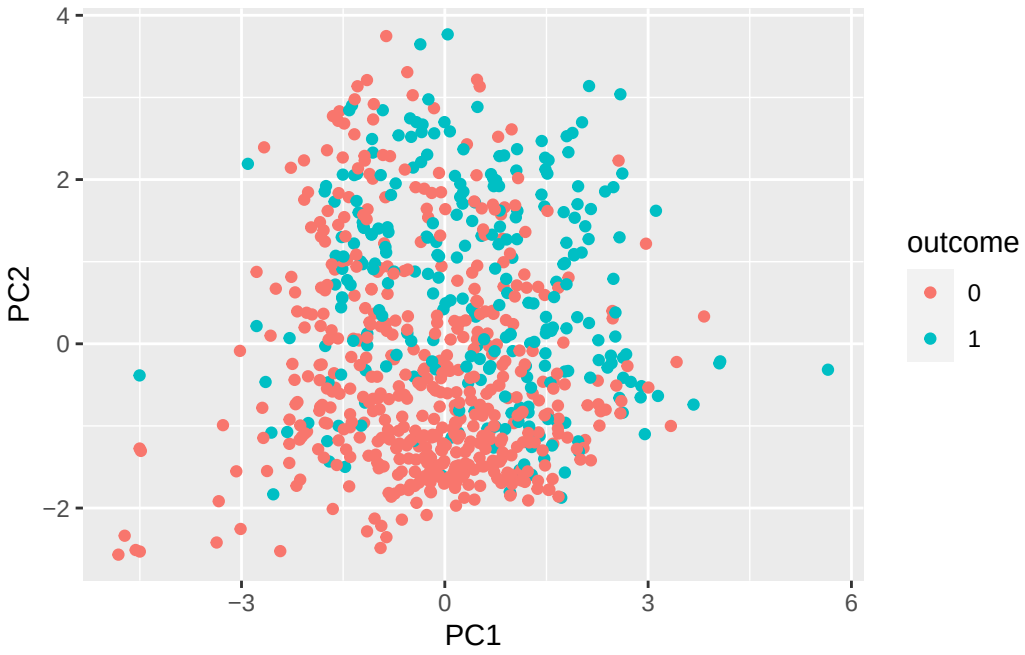
La transformación logarítmica es útil para transformar distribuciones con sesgo positivo (con cola más larga hacia la derecha): la parte izquierda se expandirá, mientras que la derecha se comprimirá, favoreciendo que la curva resultante se ajuste mejor a una normal.

```
datos$Insulin  <- log(datos$Insulin+0.05)# se suma 0.05 antes de aplicar el logaritmo con

summary(datos)# nuevamente un resumen de los 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
```

```
pcx <- prcomp(datos[,1:n1],scale. = T) ## escalamos por la variablidad de los datos
# Nuevo análisis se componentes principales.
plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)#Se combinan las columnas de los resultad
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()#nuevamente un gráfico de dispersió
```



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..

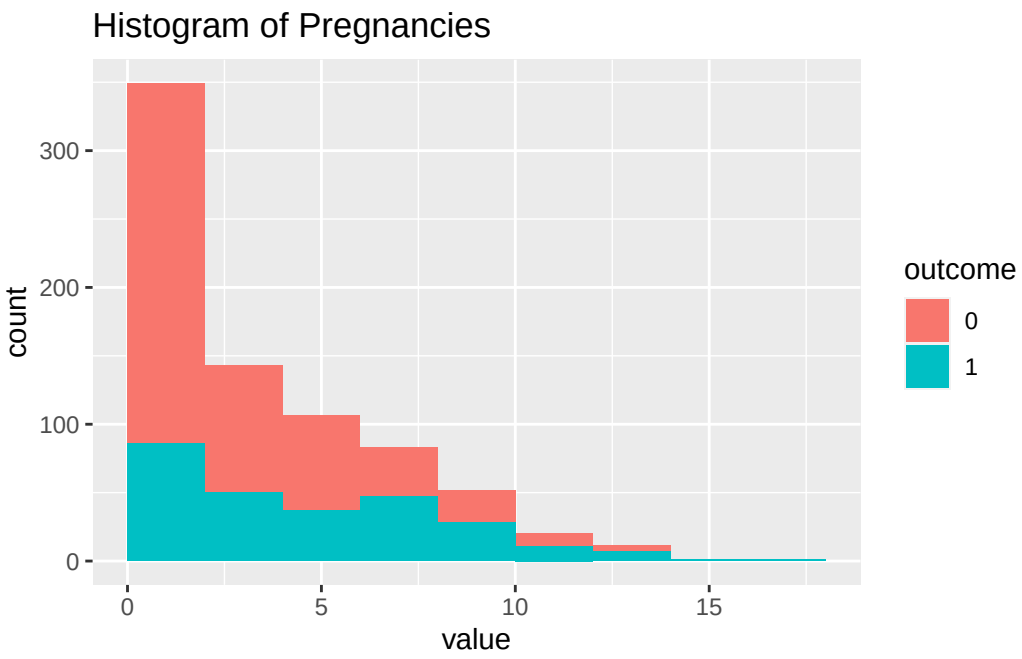
```r
datos <- read.csv("./datos/diabetes.csv")# se cargan los datos.
datos$Outcome <- as.factor(datos$Outcome)#conversión de la columna Outcome de datos en un
datsc <- scale(datos[,-ncol(datos)])# se estandarízan los datos numéricos   y se exluye la
```

Veamos las distribuciones de nuevo....

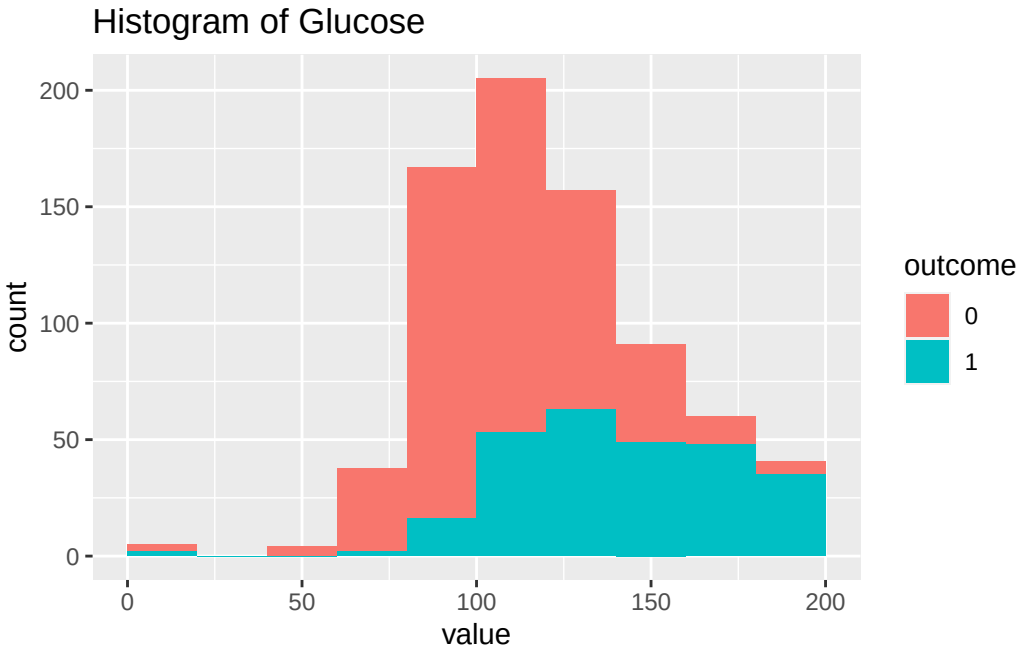
Nuevos histogramas después de la tranformación de Insulina

```r
l.plots <- vector("list",length = ncol(datos)-1)# lista vacía con una longitud igual al nú
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# guarda el gráfico de histograma generado en la posición j de la list

}
l.plots# esta lista contiene una serie de gráficos de histogramas generados para las colum
```

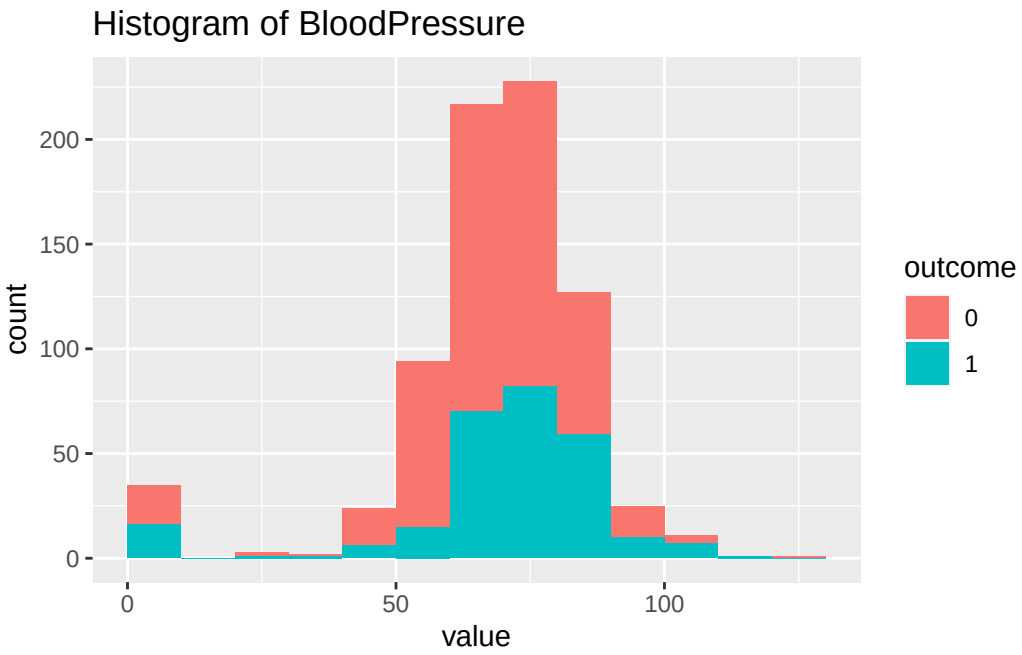
```
[[1]]
```


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

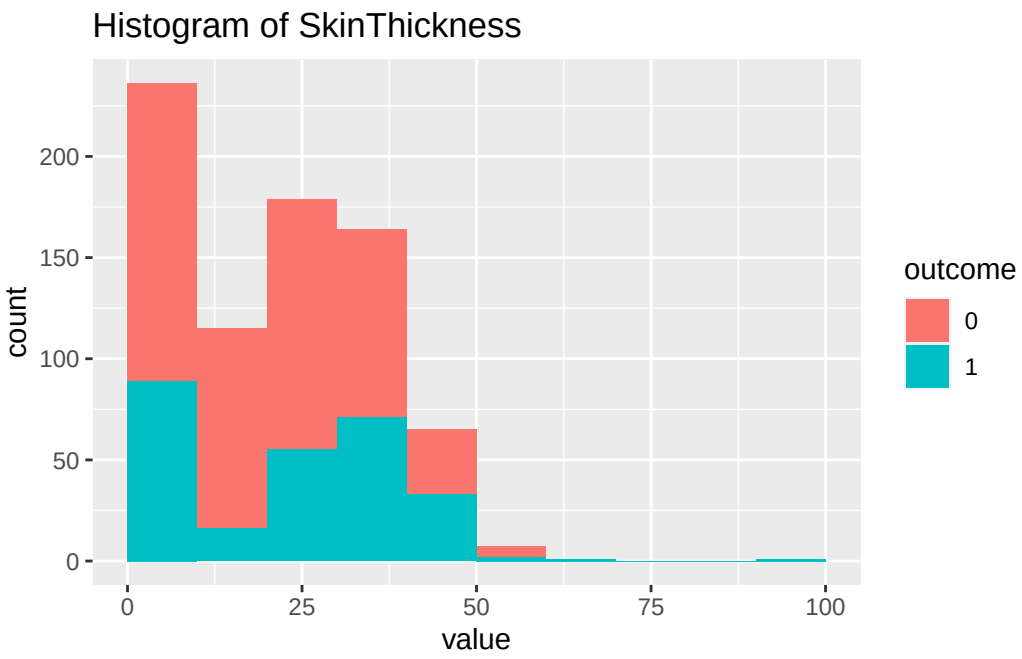
```
[[2]]
```


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

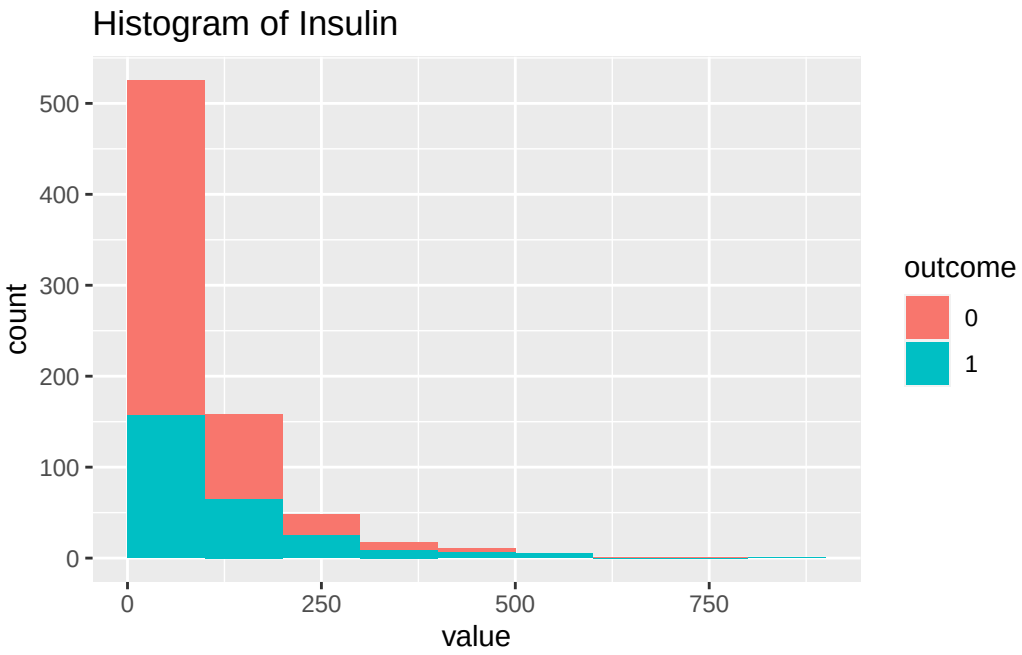
[[3]]


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

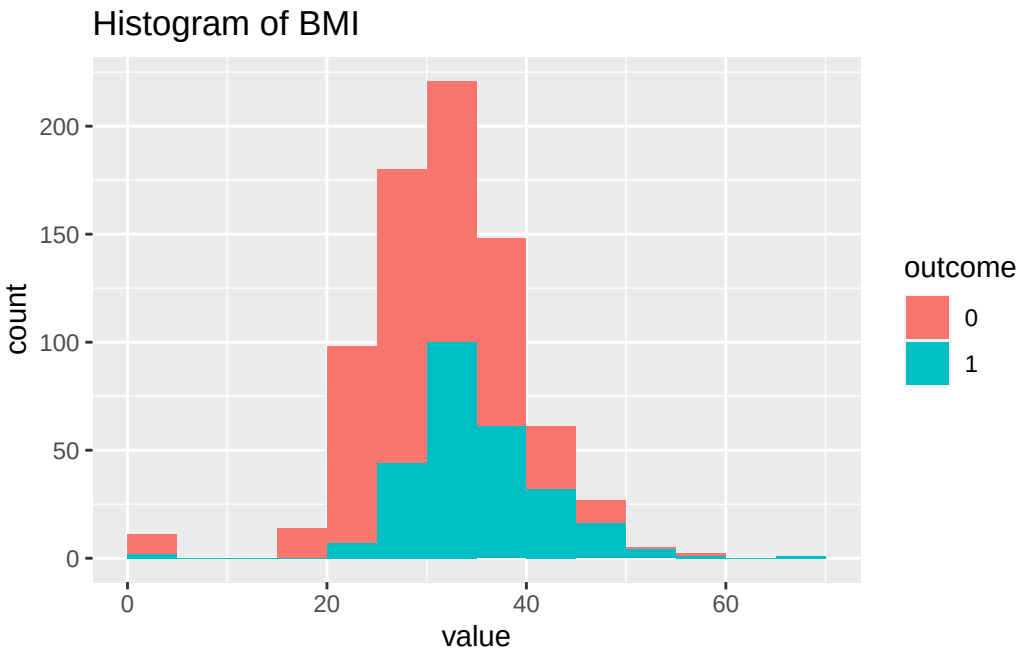
[[4]]


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

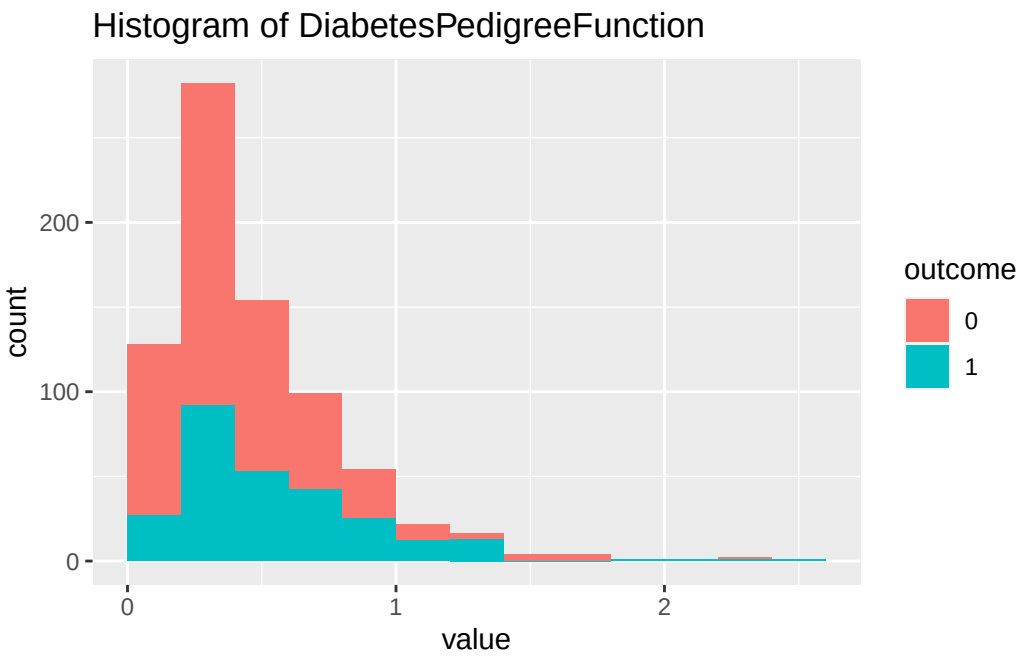
```
[[5]]
```


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

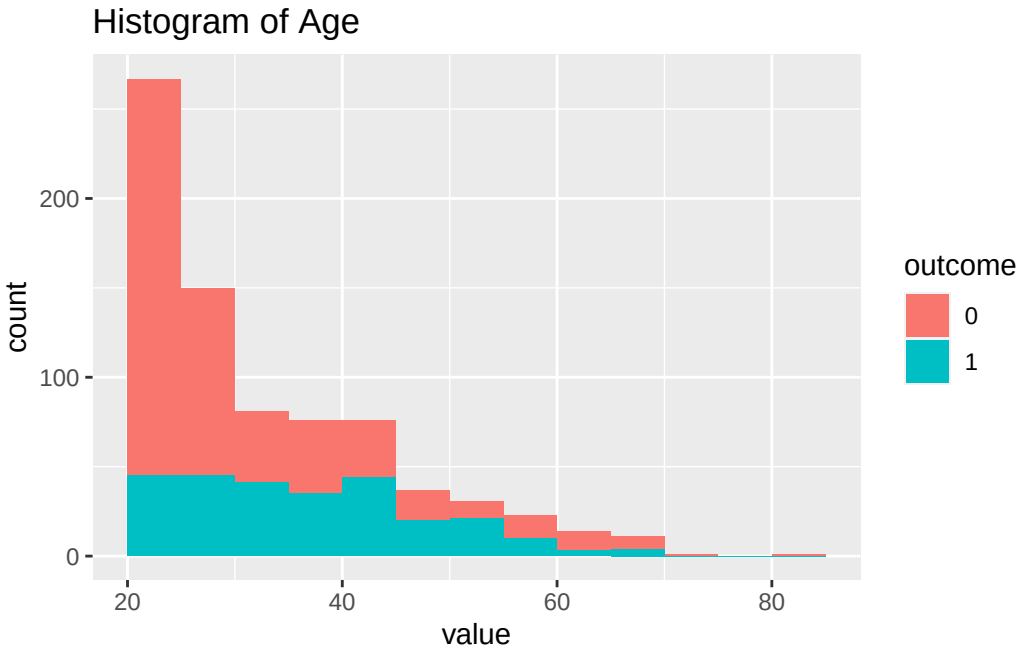
```
[[6]]
```


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]]
```

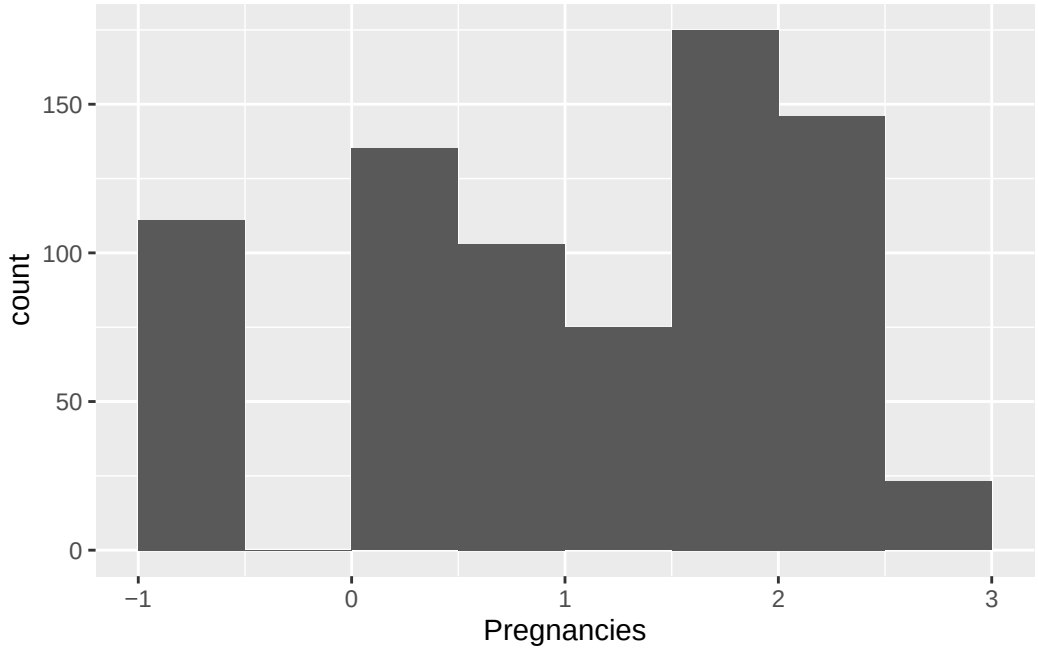

Histogram of Age
count
200
100
0
20
40
60
80
value
outcome
0
1

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...

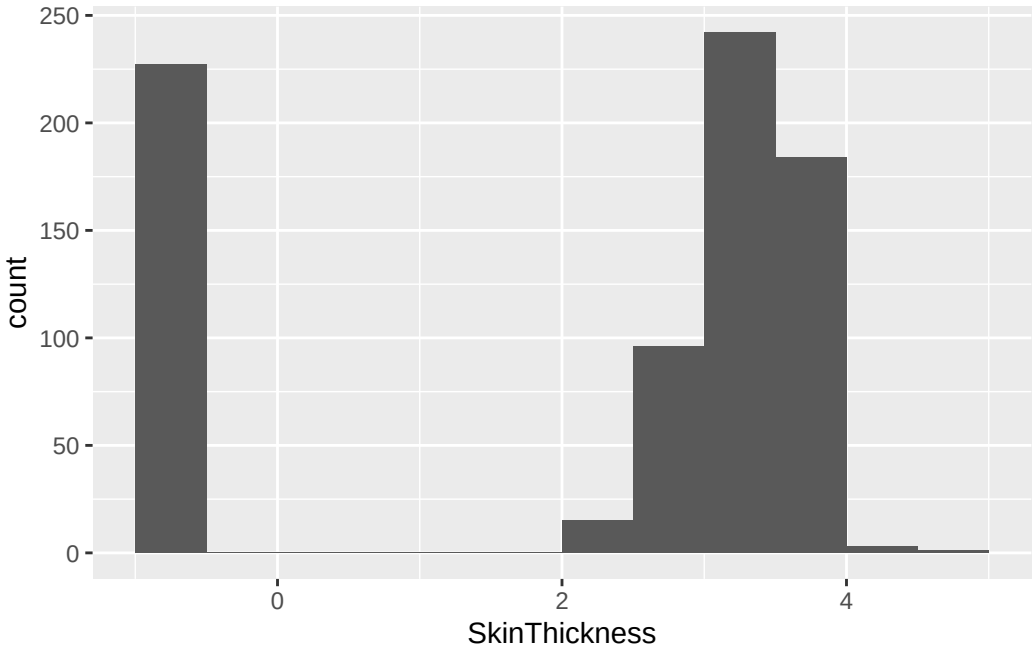
Histograma para la variable "Pregnacies"

```
datos <- read.csv("./datos/diabetes.csv")#Cargamos los datos
datos$Outcome <- as.factor(datos$Outcome)# se convierten los datos outcome a un factor.
datos$Pregnancies  <- log(datos$Pregnancies+0.5)#Se aplica la tranformación logarítmica pa
ggplot(datos,aes(Pregnancies))+geom_histogram(breaks = hist(datos$Pregnancies,plot=F)$brea
```



Realizaremos lo mismo con la grosura de la piel

```
datos <- read.csv("./datos/diabetes.csv")#Cargamos los datos
datos$Outcome <- as.factor(datos$Outcome)# se convierten los datos outcome a un factor.
datos$SkinThickness  <- log(datos$SkinThickness+0.5)#Se aplica la tranformación logarítmic
ggplot(datos,aes(SkinThickness))+geom_histogram(breaks = hist(datos$SkinThickness,plot=F)$
```



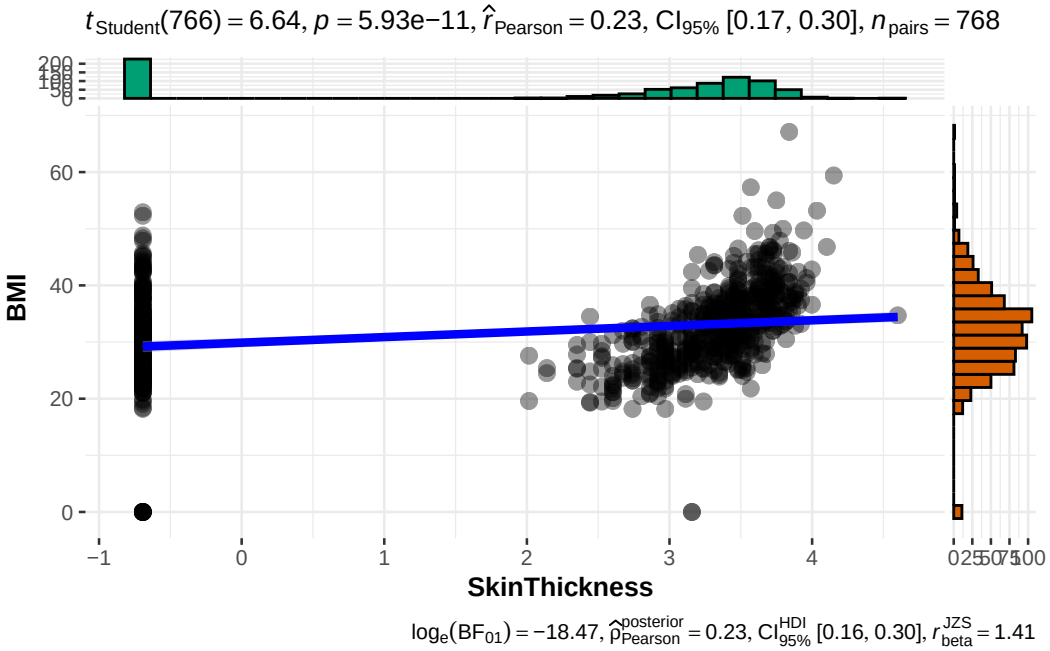
Tenemos algo raro, lo más posible sea por la obesidad…

Gráfico de dispersión y prueba estadísticas adecuadas.

Para los valriables: “SkinThickness” y “BMI”.

```
ggscatterstats(datos,SkinThickness,BMI)# esta sección realiza una comparación estadística
```

```
`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...

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

datos$Outcome <- as.factor(datos$Outcome)# Se convierte la columna outcome a un factor con
```

vamos a quitar estos valores

```
datos <- datos[complete.cases(datos),]# se eliminan todas las filas en datos. Solo se cons
```

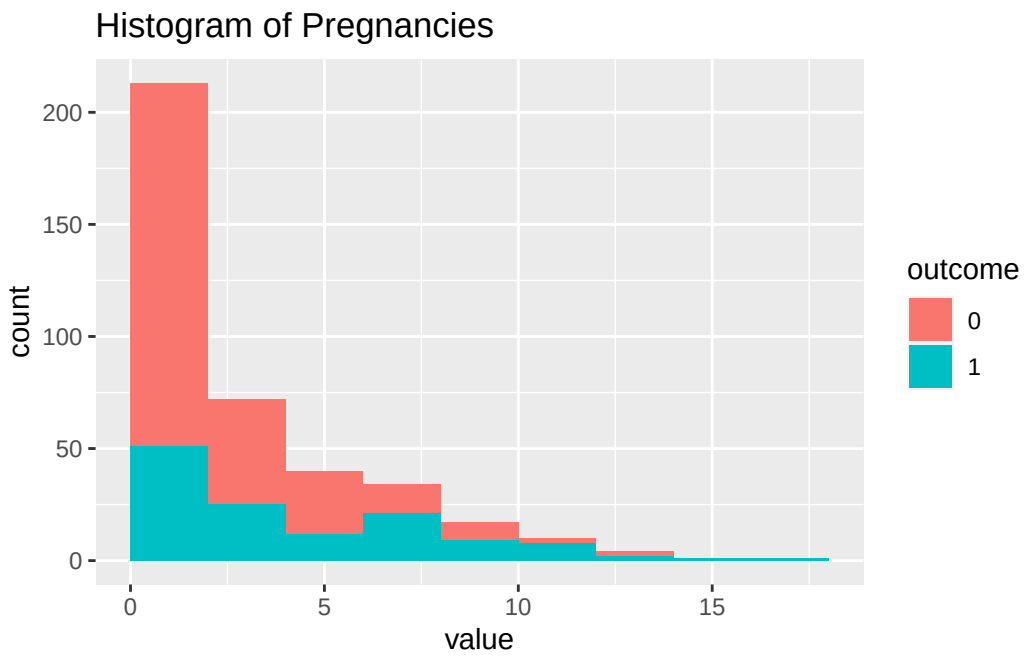
Se redujo el data set a 392 observaciones...

```
table(datos$Outcome) #Esta línea muestra una tabla de frecuencias de la variable "Outcome"
```

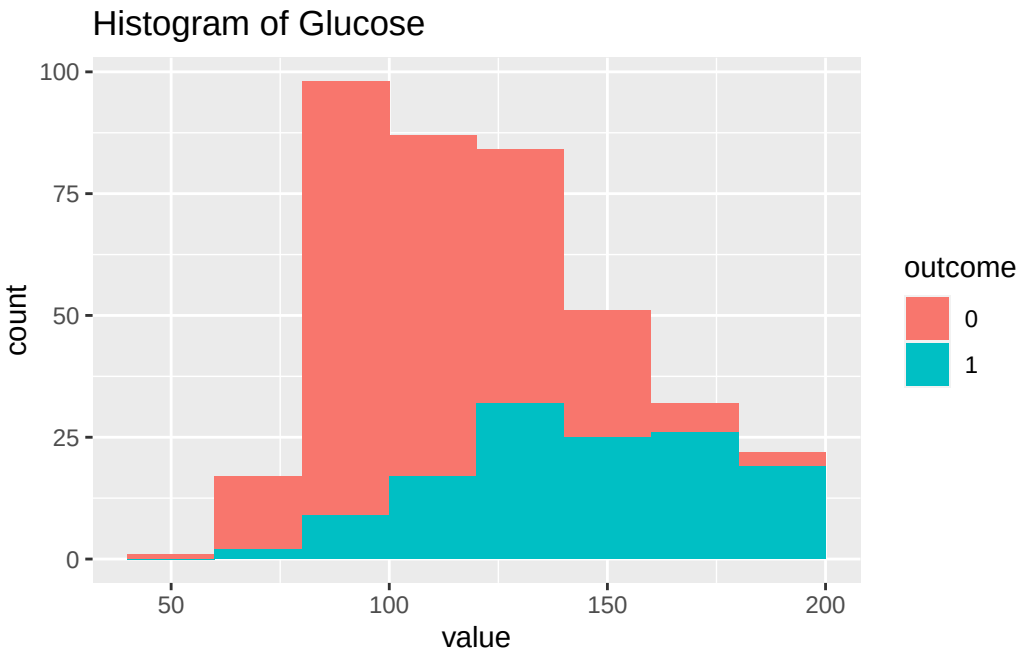
```
  0   1
262 130
```

```r
l.plots <- vector("list",length = ncol(datos)-1)#se crea una lista llamada l.plots con lon
n1 <- ncol(datos) -1
for(j in 1:n1){
  #itera sobre cada columna de datos (excepto la columna "Outcome") y realiza las siguient
  h <-hist(datos[,j],plot = F)#calcula el histograma de la columna j
  datos.tmp <- data.frame(value=datos[,j],outcome=datos$Outcome)# crea un nuevo marco de d
  p1 <- ggplot(datos.tmp,aes(value,fill=outcome))+geom_histogram(breaks=h$breaks) + ggtitl

  l.plots[[j]] <- p1
}
l.plots# se muestran los histogramas
```

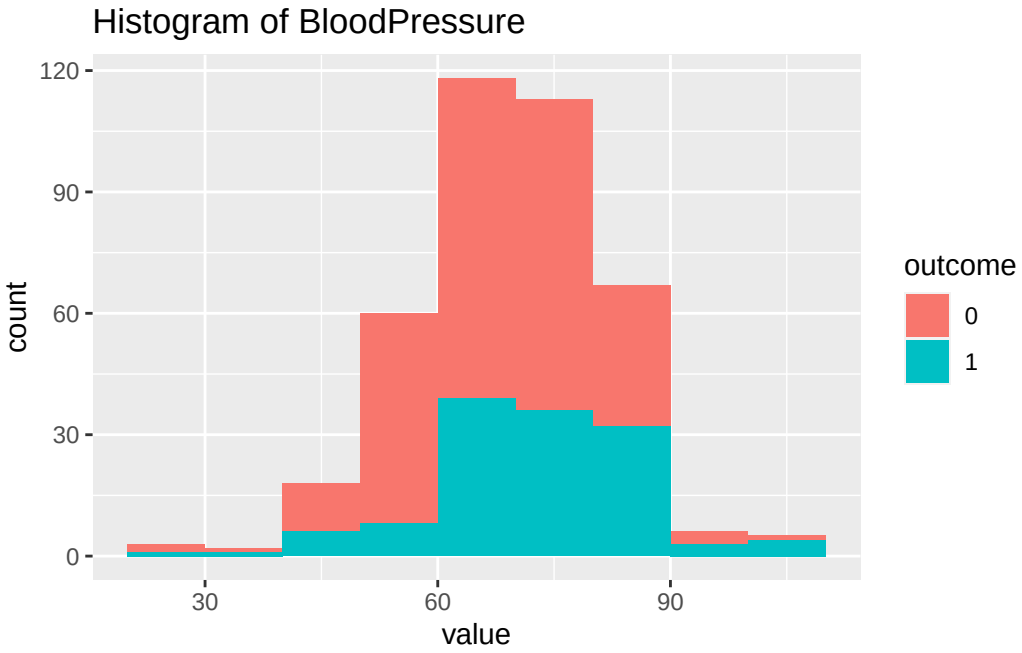
```
[[1]]
```



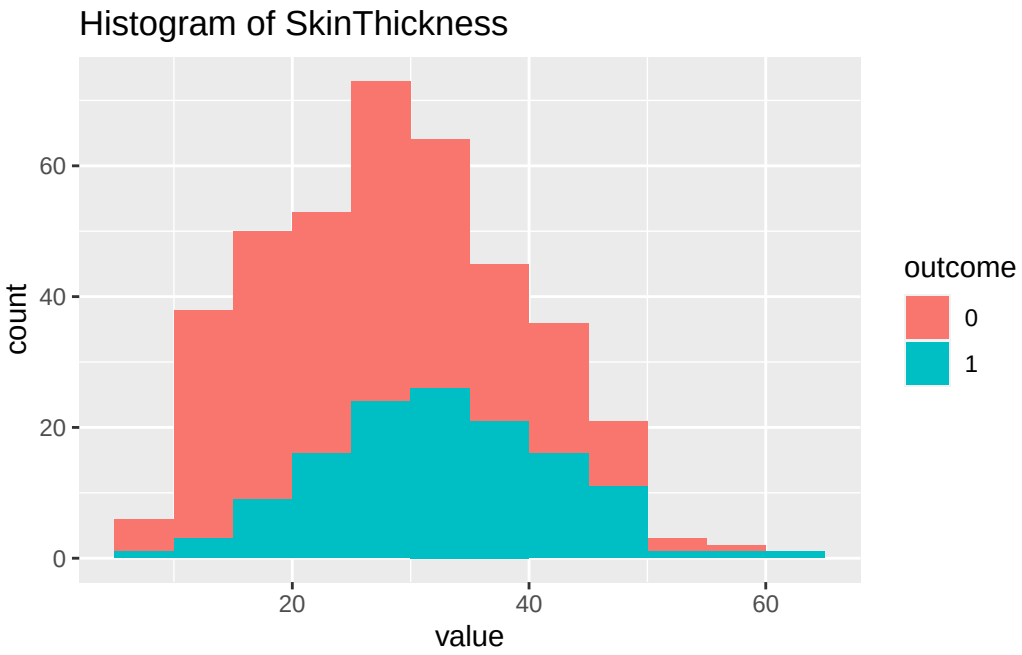

```
[[2]]
```


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

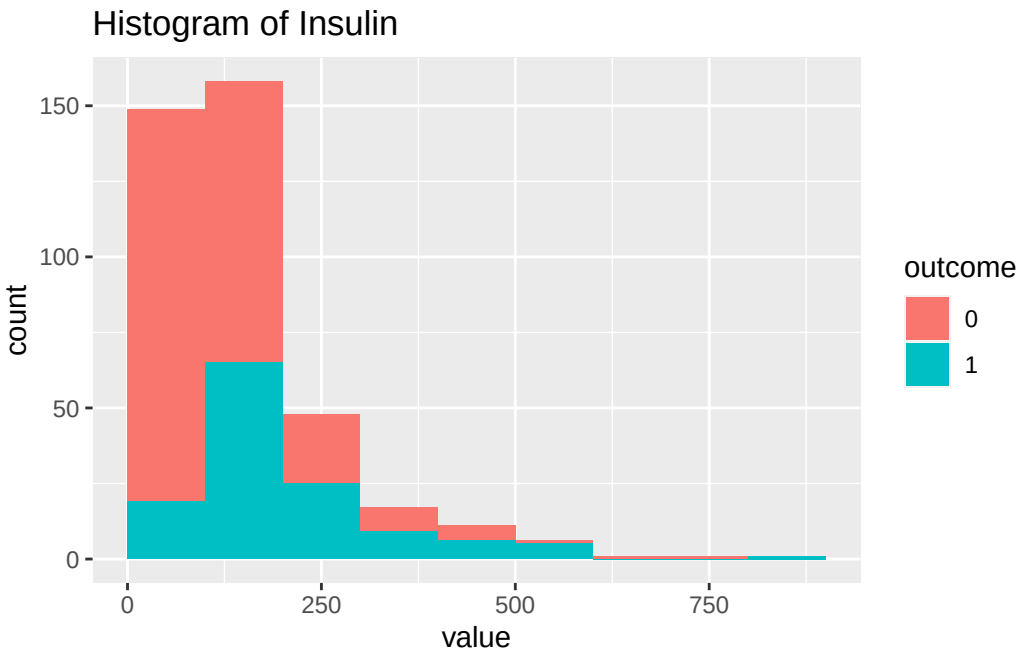
```
[[3]]
```


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

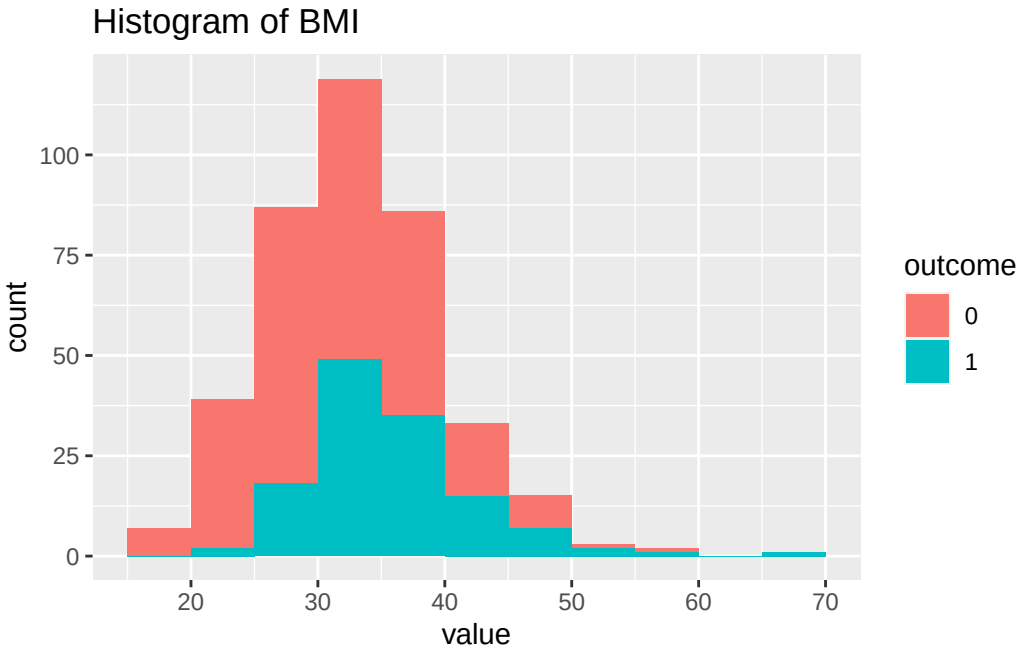
```
[[4]]
```


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

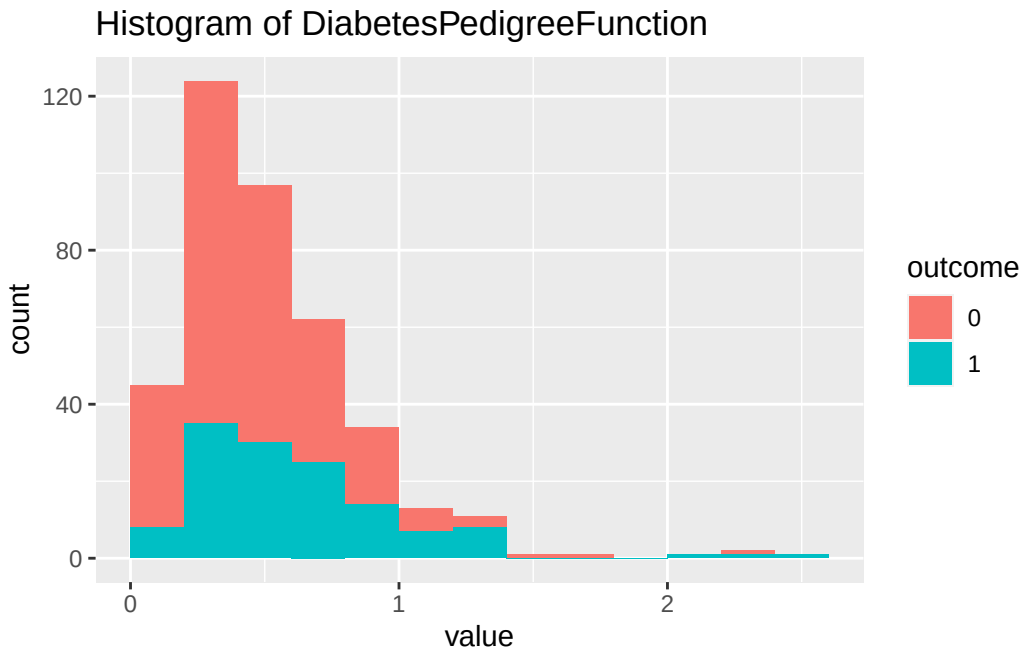
```
[[5]]
```


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

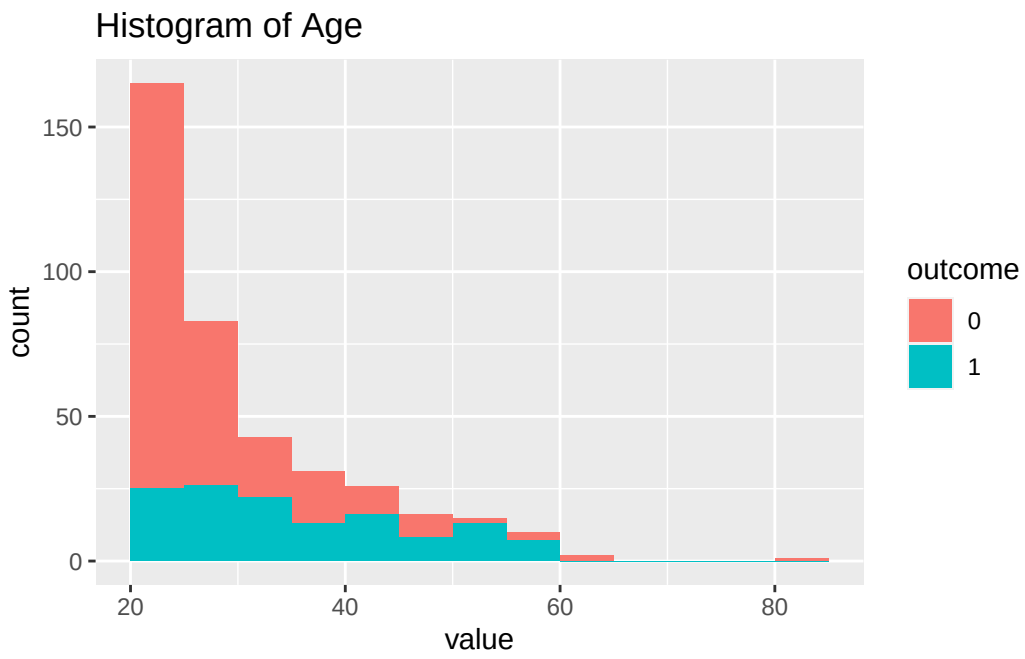
```
[[6]]
```


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

[[7]]


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

[[8]]



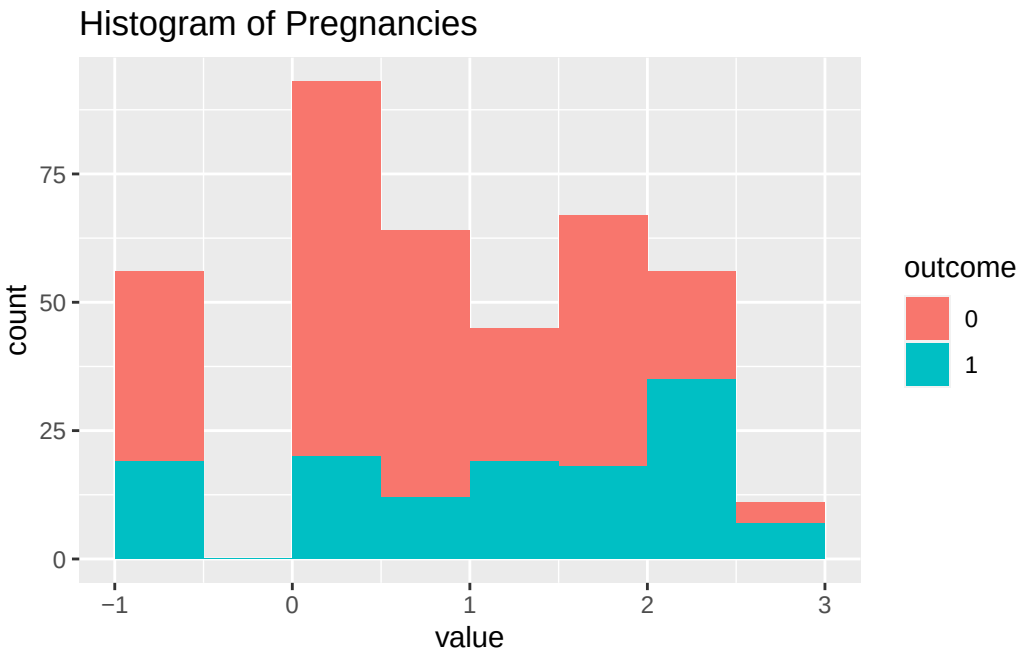
Ahora si podemos realizar las transfomraciones

Se muestran los histogramas de las transformaciones

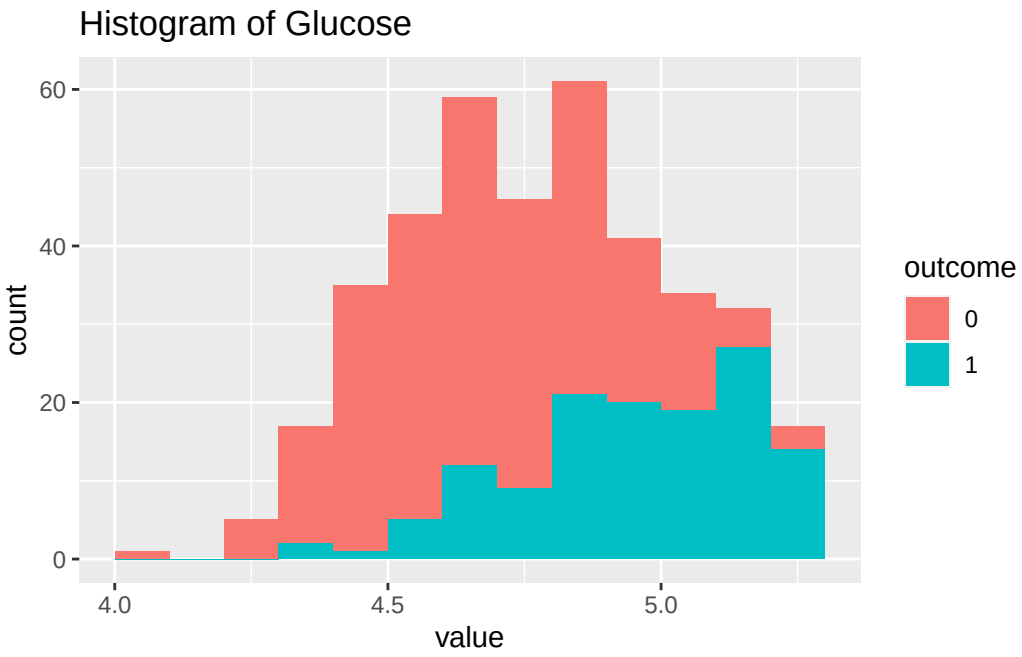
```
datos <- read.csv("./datos/diabetes.csv")# se lee el código
datos[,-c(1,9)] <- apply(datos[,-c(1,9)],2,function(x) ifelse(x==0,NA,x))# se plica una fu
datos <- datos[complete.cases(datos),]#Se eliminan todas las filas en datos que contienen

datos$Outcome <- as.factor(datos$Outcome)# se conveirte a factor
datos$Insulin <- log(datos$Insulin)#se aplica la tranformación logaritmica a Insulina
datos$Pregnancies <- log(datos$Pregnancies+0.5)#se aplica la tranformación logaritmica a p
datos$DiabetesPedigreeFunction <- log(datos$DiabetesPedigreeFunction)#se aplica la tranfor

datos$SkinThickness <- sqrt((datos$SkinThickness))#raíz cuadrada de SkinThickness
datos$Glucose <- log(datos$Glucose)# se aplica una transformación logarítmica a la columna
datos$Age <-log2(datos$Age)# Se aplica una transformación logarítmica en base 2 a la colum
l.plots <- vector("list",length = ncol(datos)-1)
n1 <- ncol(datos) -1#Esta línea guarda el número de columnas en datos menos 1 en el objeto
for(j in 1:n1){

  h <-hist(datos[,j],plot = F)#Esta línea calcula el histograma de la columna j de datos u
```

```
  datos.tmp <- data.frame(value=datos[,j],outcome=datos$Outcome)#Esta línea crea un nuevo
  p1 <- ggplot(datos.tmp,aes(value,fill=outcome))+geom_histogram(breaks=h$breaks) + ggtitl

  l.plots[[j]] <- p1#Esta línea guarda el gráfico de histograma generado en la posición j
}
l.plots # se muestran los histogramas de la lista
```

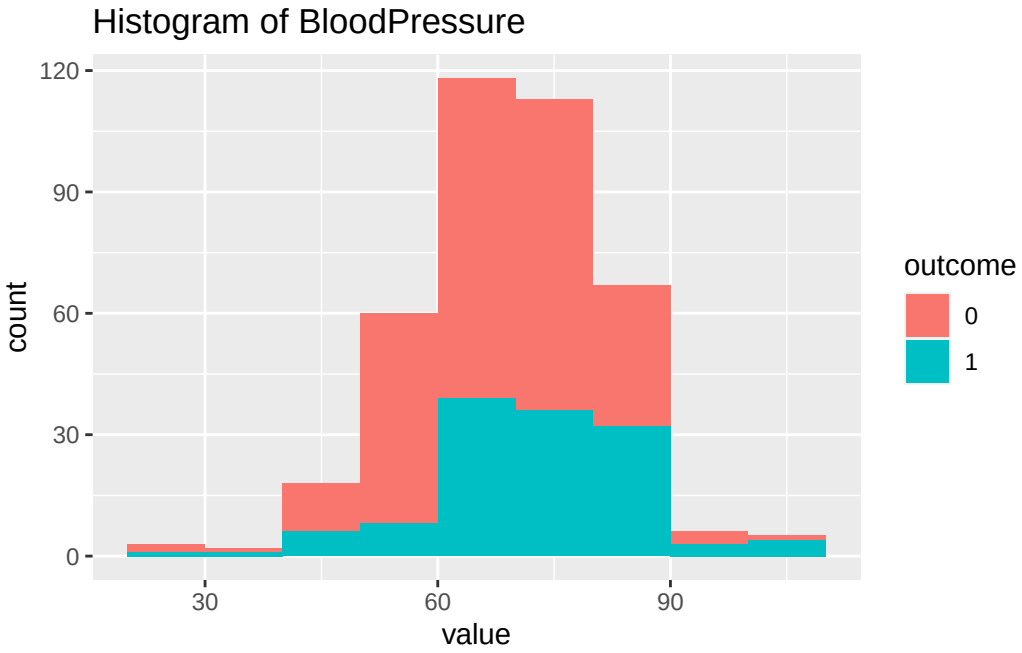
```
[[1]]
```



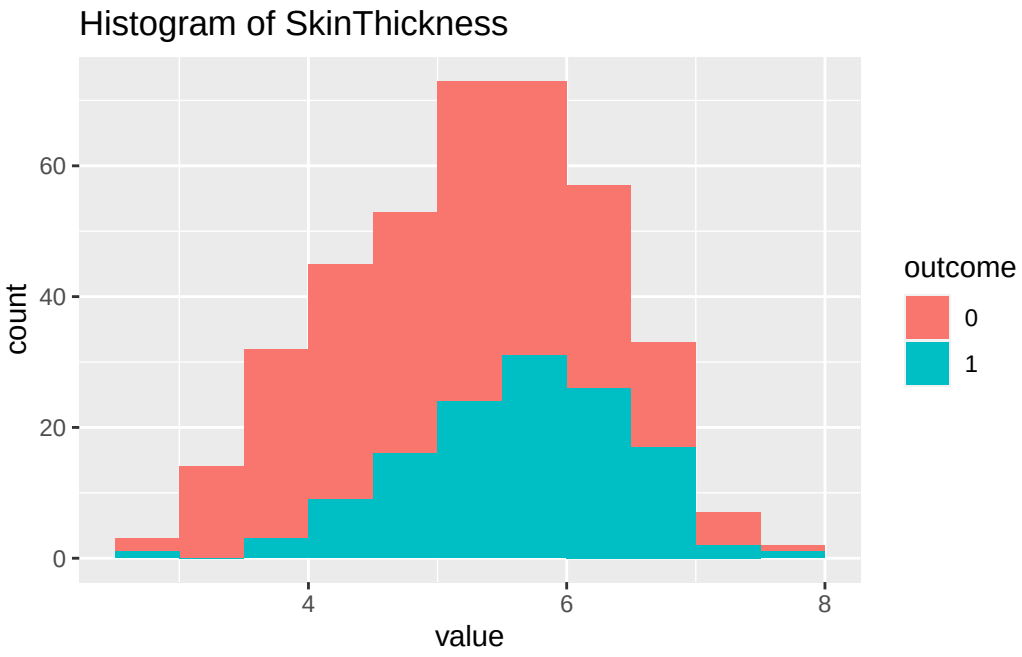

```
[[2]]
```


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

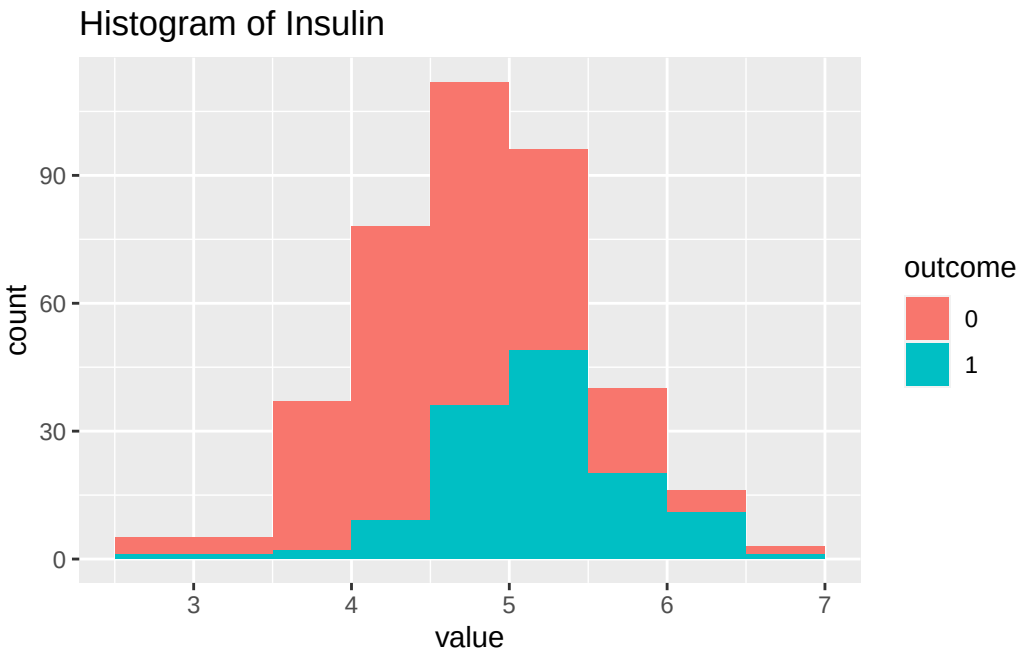
```
[[3]]
```


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

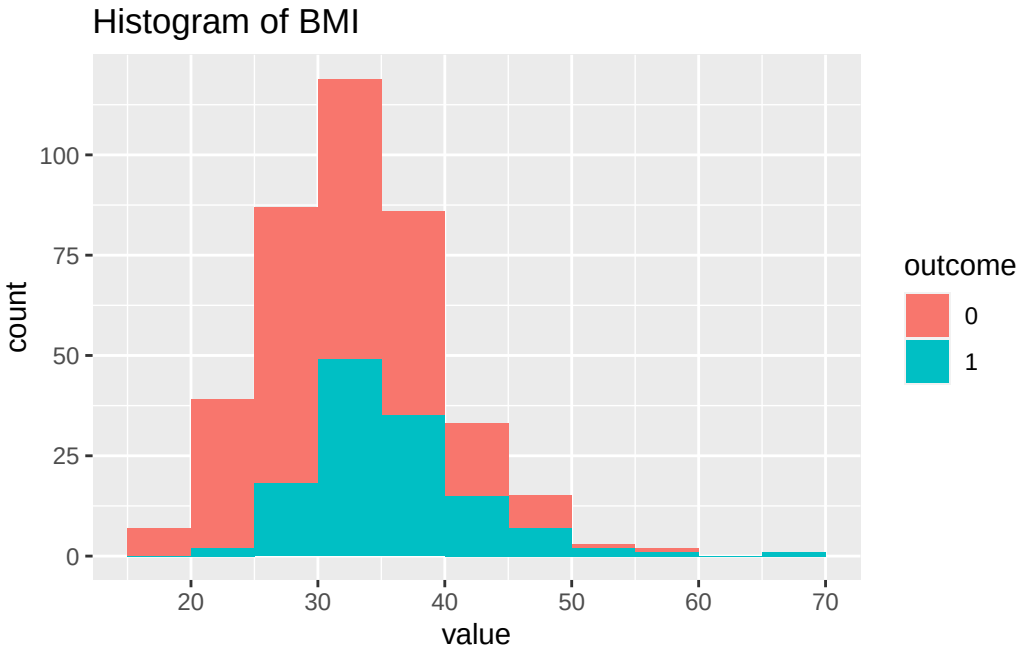
```
[[4]]
```


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

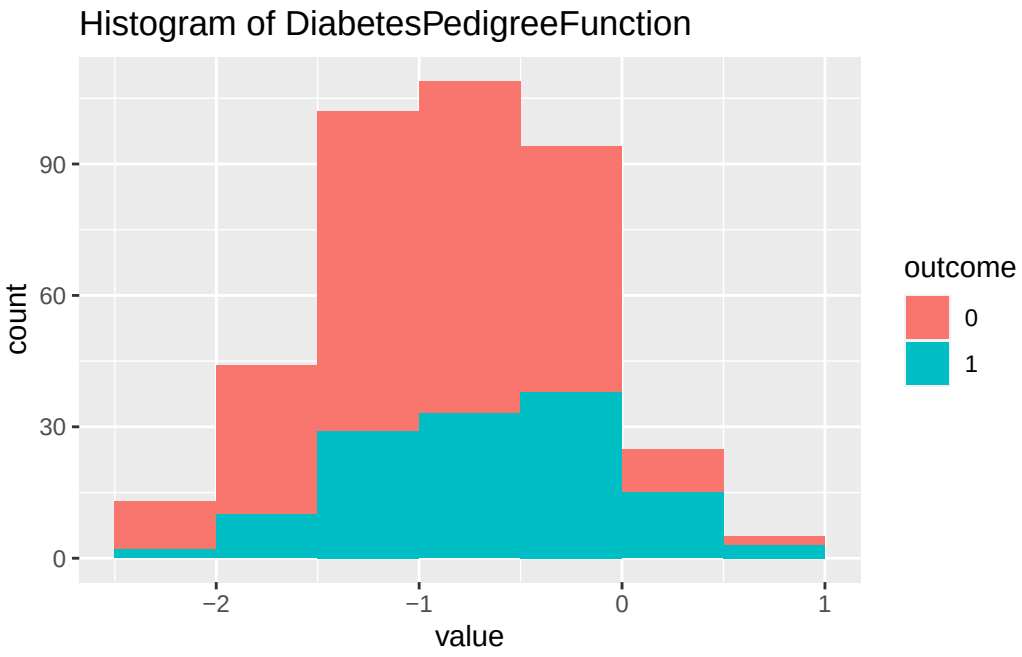
```
[[5]]
```


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

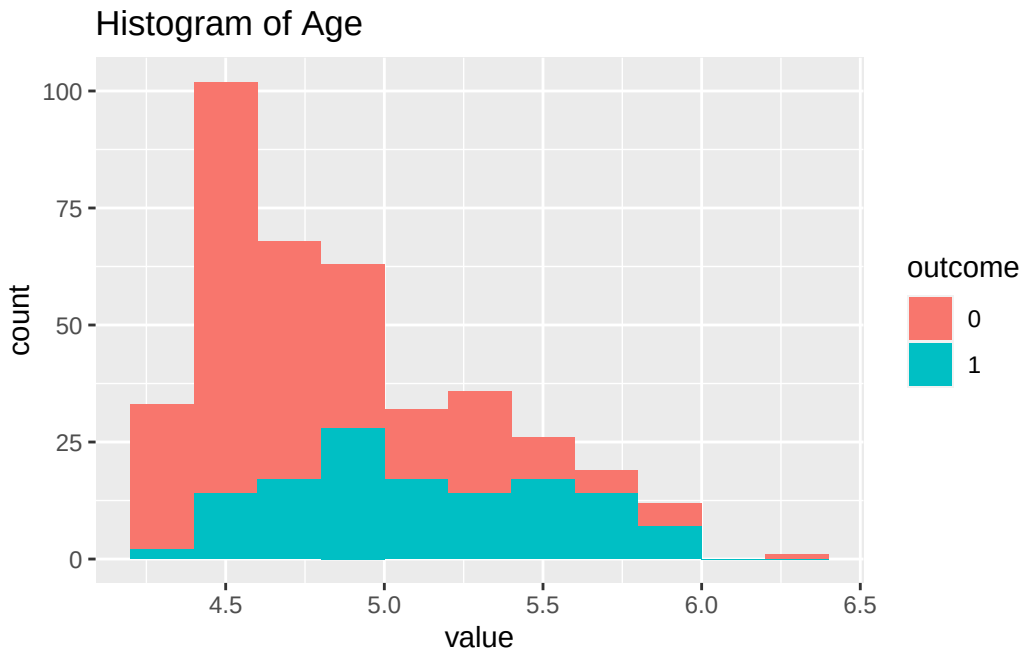
```
[[6]]
```


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]]
```



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

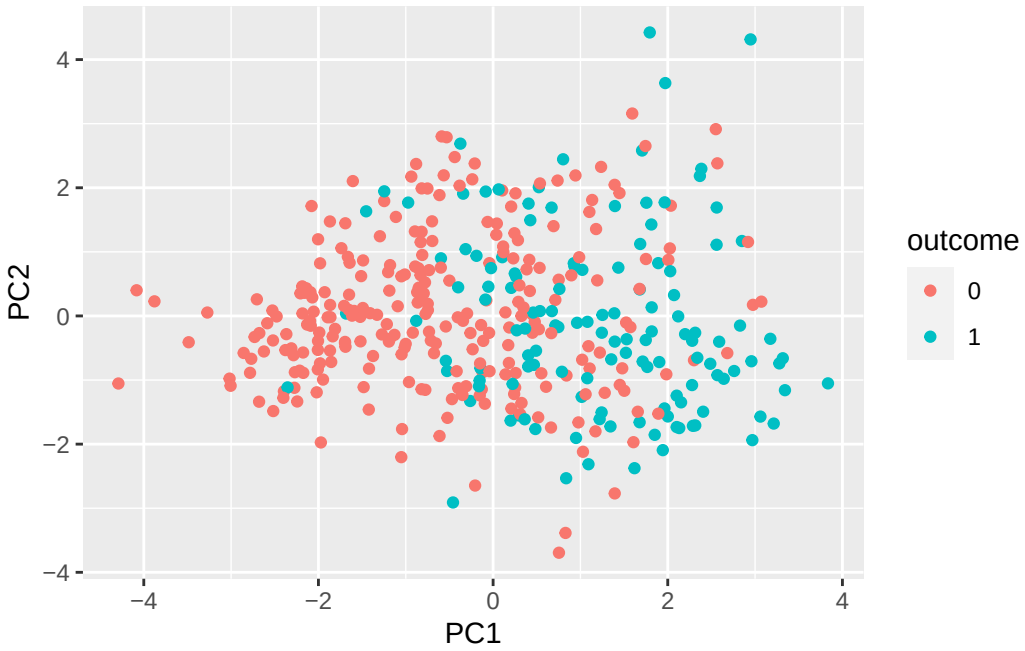
Gráfico de dispersión

```
summary(datos)# resumen de los 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
#análisis de componente principales.
plotpca <- bind_cols(pcx$x,outcome=datos$Outcome)# se brindan principales calculados por l
ggplot(plotpca,aes(PC1,PC2,color=outcome))+geom_point()# gráfico de disperción
```



Ahora vamos a realizar las pruebas de medianas

```
p.norm <- apply(apply(scale(datos[,1:n1]), #Se  línea realiza una prueba de normalidad de
            2,
            function(x) summary(lm(x~datos$Outcome))$residuals),
      2,
      shapiro.test)#shapiro.test() se utiliza para realizar la prueba de normalidad.

p.norm# Se  muestra los resultados de las pruebas de normalidad realizadas en el paso ante
```

```
$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

Prueba de Wilcoxon-Mann-Whitney

Este mpetodo se utiliza con frecuencia para comparar medias o medianas de dos conjuntos independientes, posiblemente con distribución no normal.

```r
p.norm <- apply(scale(datos[,1:n1]),
            2,
            function(x) wilcox.test(x~datos$Outcome)$p.value)
# Esta línea realiza una prueba de Wilcoxon-Mann-Whitney para cada columna (excepto la col
```

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

Ajuste delos valores de p resultantes de las pruebas de normalidad utilizando el método de ajuste de Benjamini-Hochberg

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

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

```
datos.split <- split(datos,datos$Outcome)# Se divide el conjunto de datos datos en subconj

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


toplot <- data.frame(medianas=Reduce("-",datos.median)
,p.values=p.adj)#combina las medianas calculadas en el paso anterior y los valores de p aj

toplot# Se muestra el marco de datos toplot, que contiene las medianas de cada columna y l
```

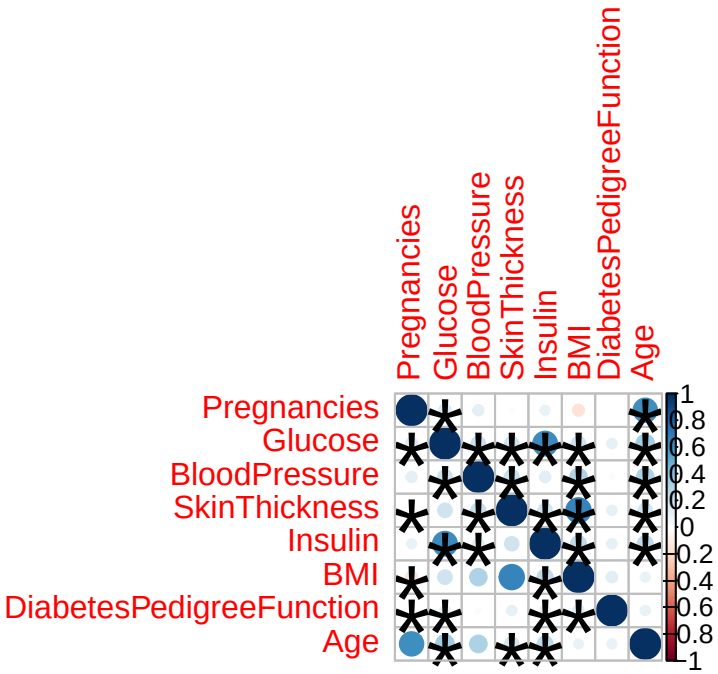
```
                           medianas     p.values
Pregnancies               -0.3364722 8.957407e-05
Glucose                   -0.2957935 4.902429e-22
BloodPressure             -4.0000000 8.957407e-05
SkinThickness             -0.5484102 4.309442e-07
Insulin                   -0.4788534 3.241934e-13
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

Método de ajuste de Benjamini-Hochberg.

Este método asume a la hora de estimar el número de hipótesis nulas erróneamente consideradas falsas, que todas las hipótesis nulas son ciertas.

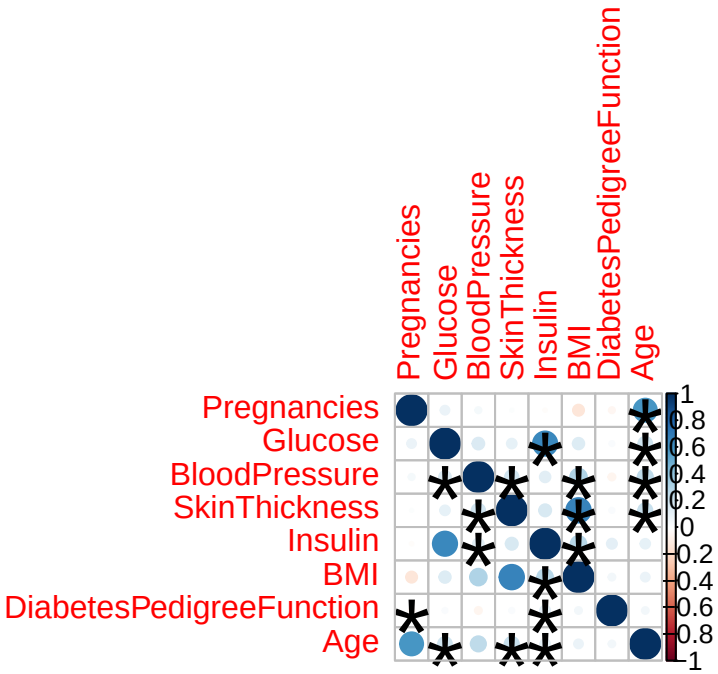
```
obj.cor <- psych::corr.test(datos[,1:n1])#Se realiza un análisis de correlación en las col
p.values <- obj.cor$p# Se realiza un ajuste de los valores de p resultantes de las pruebas
p.values[upper.tri(p.values)] <- obj.cor$p.adj
p.values[lower.tri(p.values)] <- obj.cor$p.adj
diag(p.values) <- 1
corrplot::corrplot(corr = obj.cor$r,p.mat = p.values,sig.level = 0.05,insig = "label_sig")
```



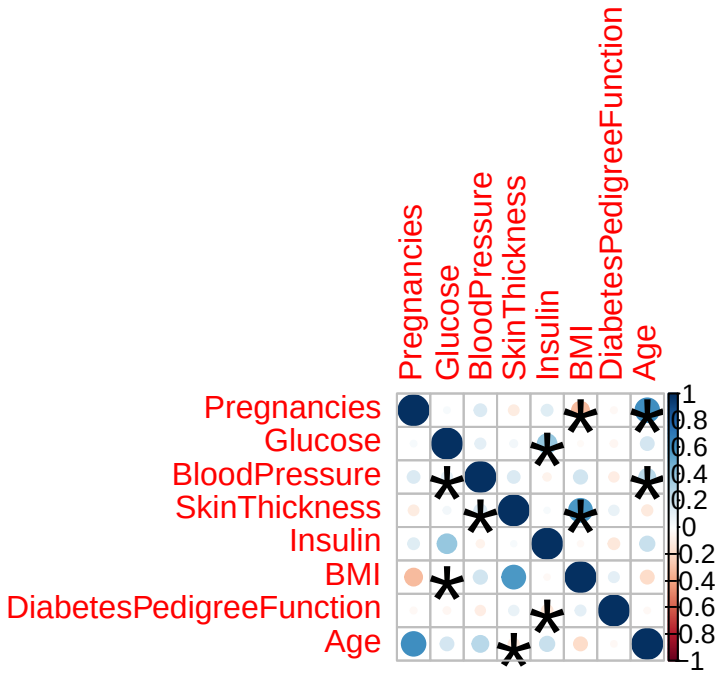
También podemos observar como cambian las relaciones segun la diabetes

Estas líneas realizan análisis de correlación separados para los subconjuntos de datos donde el valor de "Outcome" es igual a 0 y 1, respectivamente. Se calculan las matrices de correlación y se ajustan los valores de p utilizando el método de Benjamini-Hochberg.

```
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

Estandarización de las variables predictoras.

El objetivo es escoger el 70 porciento de los datos para el conjunto de entrenamiento y el otro 30 para la prueba.

```
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))# Se asignan los valores estanda
levels(datos$Outcome) <- c("D","N")# se revienreten los valores de la variable
train <- sample(nrow(datos),size = nrow(datos)*0.7)# se selecciona una muestra de un 70 %

dat.train <- datos[train,]# datos de entrenamiento
dat.test <- datos[-train,]# datos de prueba.
```

Modelado

Modelo de regresión logística. Este modelo se ajusta utilizando la variable objetivo "Outcome" en función de todas las demás variables predictoras

```
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"))# S
caret::confusionMatrix(prediccion,dat.test$Outcome)#matriz de confusión
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 74 12
         N  9 23

               Accuracy : 0.822
                 95% CI : (0.7409, 0.8863)
    No Information Rate : 0.7034
    P-Value [Acc > NIR] : 0.002291

                  Kappa : 0.5627

 Mcnemar's Test P-Value : 0.662521

            Sensitivity : 0.8916
            Specificity : 0.6571
         Pos Pred Value : 0.8605
         Neg Pred Value : 0.7187
             Prevalence : 0.7034
         Detection Rate : 0.6271
   Detection Prevalence : 0.7288
      Balanced Accuracy : 0.7744

       'Positive' Class : D
```

LASSO

La regresión logística de Lasso es método de análisis de regresión que realiza selección de variables y regularización para **mejorar la exactitud e interpretabilidad del modelo estadístico producido por este**.

```
tuneGrid=expand.grid(
              .alpha=0,
              .lambda=seq(0, 1, by = 0.001))# Aqupi se definen la cuadrícula de hiperparám
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 76 18
         N  7 17

               Accuracy : 0.7881
                 95% CI : (0.7033, 0.858)
    No Information Rate : 0.7034
    P-Value [Acc > NIR] : 0.02508

                  Kappa : 0.4415

 Mcnemar's Test P-Value : 0.04550

            Sensitivity : 0.9157
            Specificity : 0.4857
         Pos Pred Value : 0.8085
         Neg Pred Value : 0.7083
             Prevalence : 0.7034
         Detection Rate : 0.6441
   Detection Prevalence : 0.7966
```

```
      Balanced Accuracy : 0.7007

       'Positive' Class : D
```

Estas líneas entrena el modelo de regresión logística regularizado utilizando la función **train()** del paquete caret. Se utiliza el método "glmnet" y se especifican la cuadrícula de hiperparámetros y las configuraciones de control del entrenamiento definidas anteriormente. La métrica de evaluación utilizada es la precisión ("Accuracy").

NAVIE BAYES

Aprende de los datos de entrenamiento y luego predice la clase de la instancia de prueba con la mayor probabilidad posterior. También es útil para datos dimensionales altos ya que la probabilidad de cada atributo se estima independientemente.

```
tuneGrid=expand.grid(
              .alpha=1,
              .lambda=seq(0, 1, by = 0.0001))
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 74 16
         N  9 19

               Accuracy : 0.7881
                 95% CI : (0.7033, 0.858)
```

```
    No Information Rate : 0.7034
    P-Value [Acc > NIR] : 0.02508

                  Kappa : 0.4611

 Mcnemar's Test P-Value : 0.23014

            Sensitivity : 0.8916
            Specificity : 0.5429
         Pos Pred Value : 0.8222
         Neg Pred Value : 0.6786
             Prevalence : 0.7034
         Detection Rate : 0.6271
   Detection Prevalence : 0.7627
      Balanced Accuracy : 0.7172

       'Positive' Class : D
```

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

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 61 16
         N 11 30

               Accuracy : 0.7712
                 95% CI : (0.6848, 0.8435)
    No Information Rate : 0.6102
    P-Value [Acc > NIR] : 0.0001537
```

```
                 Kappa : 0.5094

 Mcnemar's Test P-Value : 0.4414183

            Sensitivity : 0.8472
            Specificity : 0.6522
         Pos Pred Value : 0.7922
         Neg Pred Value : 0.7317
             Prevalence : 0.6102
         Detection Rate : 0.5169
   Detection Prevalence : 0.6525
      Balanced Accuracy : 0.7497

       'Positive' Class : D
```

```
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])
```

Para esta seccion se elecciona el valor de lambda mínimo del modelo regularizado. Se busca el valor de lambda que es mayor o igual que el valor de lambda óptimo seleccionado durante el ajuste del modelo.

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

```
[1] "(Intercept)"              "Glucose"
[3] "Insulin"                  "BMI"
[5] "DiabetesPedigreeFunction" "Age"
```

```
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)# Esta línea calcula la matriz de confusión pa
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
```

```
         D 61 19
         N 11 27

               Accuracy : 0.7458
                 95% CI : (0.6574, 0.8214)
    No Information Rate : 0.6102
    P-Value [Acc > NIR] : 0.001351

                  Kappa : 0.4483

 Mcnemar's Test P-Value : 0.201243

            Sensitivity : 0.8472
            Specificity : 0.5870
         Pos Pred Value : 0.7625
         Neg Pred Value : 0.7105
             Prevalence : 0.6102
         Detection Rate : 0.5169
   Detection Prevalence : 0.6780
      Balanced Accuracy : 0.7171

       'Positive' Class : D
```

Este seccion nos ayuda a crear un objeto **trainControl** que define la configuración para el entrenamiento del modelo. En este caso, se utiliza validación cruzada repetida como método de validación.

```r
library(ISLR)
library(caret)
set.seed(400)#semilla
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#Muestra el resultado del ajuste del modelo k-NN, que incluye información sobre los
```

```
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: 247, 246, 246, 247, 246, 247, ...
Resampling results across tuning parameters:
```

k	Accuracy	Kappa
5	0.7847443	0.4663712
7	0.7848765	0.4624940
9	0.7774250	0.4319101
11	0.7811287	0.4450224
13	0.7859347	0.4554923
15	0.7748236	0.4222221
17	0.7750882	0.4184350
19	0.7676808	0.3984064
21	0.7725309	0.4141544
23	0.7689594	0.3981920
25	0.7640653	0.3814599
27	0.7566578	0.3534541
29	0.7566578	0.3543315
31	0.7518519	0.3365607
33	0.7591270	0.3477391
35	0.7626984	0.3646737
37	0.7602734	0.3583042
39	0.7652557	0.3653731
41	0.7590388	0.3444785
43	0.7700176	0.3740790
45	0.7664021	0.3667401
47	0.7677249	0.3670672
49	0.7627425	0.3505976
51	0.7615520	0.3497517
53	0.7652998	0.3599138
55	0.7641534	0.3551015
57	0.7665344	0.3585366
59	0.7676367	0.3601768
61	0.7639330	0.3484791
63	0.7664021	0.3545869
65	0.7700176	0.3629630
67	0.7712522	0.3590664
69	0.7687831	0.3460227
71	0.7688713	0.3446020
73	0.7689153	0.3423649
75	0.7640653	0.3282004

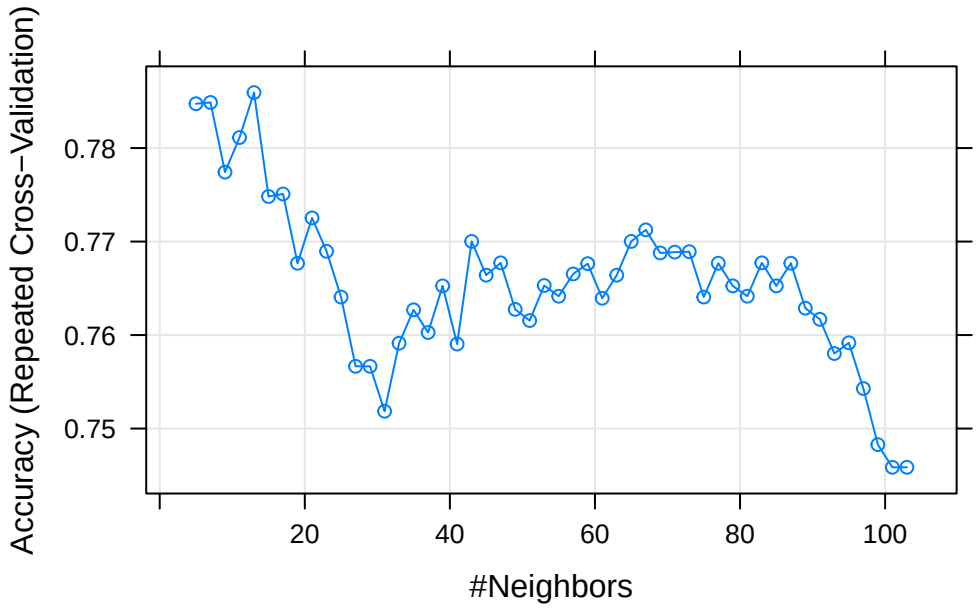
```
  77  0.7676808  0.3363602
  79  0.7652557  0.3289593
  81  0.7641534  0.3253640
  83  0.7677249  0.3348421
  85  0.7652557  0.3265140
  87  0.7676808  0.3327458
  89  0.7628748  0.3167525
  91  0.7616843  0.3104453
  93  0.7580247  0.2980541
  95  0.7591711  0.2962180
  97  0.7542769  0.2764841
  99  0.7482804  0.2541673
 101  0.7458554  0.2449811
 103  0.7458554  0.2447474

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 13.
```

Creación de un gráfico para visualizar los resultados del ajuste del modelo k-NN.

Gráfico de: ACURRACY

Para conocer el porcentaje de casos que el modelo ha acertado

```
plot(knnFit)
```



PREDICCION KNN y elaboración de su respectiva matriz de confusión

```
knnPredict <- predict(knnFit,newdata = dat.test[,-ncol(dat.test)] )#Realiza predicciones u
#Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(knnPredict, dat.test$Outcome )#Calcula la matriz de confusión para evaluar
```

```
Confusion Matrix and Statistics

          Reference
Prediction  D  N
         D 67 28
         N  5 18

               Accuracy : 0.7203
                 95% CI : (0.6302, 0.799)
    No Information Rate : 0.6102
    P-Value [Acc > NIR] : 0.0081685

                  Kappa : 0.3538

 Mcnemar's Test P-Value : 0.0001283
```

```
            Sensitivity : 0.9306
            Specificity : 0.3913
         Pos Pred Value : 0.7053
         Neg Pred Value : 0.7826
             Prevalence : 0.6102
         Detection Rate : 0.5678
   Detection Prevalence : 0.8051
      Balanced Accuracy : 0.6609

       'Positive' Class : D
```

Escala las variables predictoras en el conjunto de datos. El objetivo es crear el conjunto de datos de entrenamiento utilizando las filas seleccionadas que corresponden al 70% del conjunto de datos.

```
library(caret)
datos <- read.csv("./datos/diabetes.csv")# se leen los datos
datos$Outcome <-as.factor(datos$Outcome)# se tranfoma a factor los datos
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))#Escala las variables predictora
levels(datos$Outcome) <- c("D","N")# Define los niveles de la variable Outcome
train <- sample(nrow(datos),size = nrow(datos)*0.7)

dat.train <- datos[train,]#entrenamiento
dat.test <- datos[-train,]#testeo
set.seed(1001) #semilla
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.
```

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

confusionMatrix(prediccion,dat.test$Outcome)#matriz de confusión
```

```
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 objetivo es crear un vector **lambda** que va desde 0 hasta 50 con incrementos de 0.1. Para luego ajustar un modelo Naive Bayes y realizar predicciones.

```
datos <- read.csv("./datos/diabetes.csv")# se leen los datos
datos$Outcome <-as.factor(datos$Outcome)# se cambian los datos a factor.
levels(datos$Outcome) <- c("D","N") #Define los niveles de la variable "Outcome" como "D"
train <- sample(nrow(datos),size = nrow(datos)*0.7)# se toma el 70 porciento de los datos.

dat.train <- datos[train,]#entrenamiento
dat.test <- datos[-train,]#testeo
lambda <- seq(0,50,0.1)#Crea un vector lambda que va desde 0 hasta 50 con incrementos de 0

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

  predicciones <- predict(modelo,dat.test[,-ncol(datos)])#predicciones utilizando el model

confusionMatrix(predicciones,dat.test$Outcome)$overall[1]#Matriz de confusión.
```

```
 Accuracy
0.7705628
```

Modelo PLS-DA (Partial Least Squares Discriminant Analysis)

Este modelo tiene relación con la regresión de componentes principales, en lugar de encontrar hiperplanos de máxima varianza entre la variable de respuesta y las variables independientes, se encuentra una regresión lineal mediante la proyección de las variables de predicción y las variables observables a un nuevo espacio.

```
datos <- read.csv("./datos/diabetes.csv")
datos$Outcome <-as.factor(datos$Outcome)
datos[,1:n1] <- as.data.frame(scale(datos[,-ncol(datos)]))
```

```r
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
```

Análisis de variaza multivariante (MANOVA) sobre matrices de disimilaridad o similitud.

Este método proporciona un análisis de regresión y un análisis de varianza para variables dependientes múltiples por una o más covariables o variables de factor. Las variables de factor dividen la población en grupos. Utilizando este procedimiento de modelo lineal general, es posible contrastar hipótesis nulas sobre los efectos de las variables de factor sobre las medias de varias agrupaciones de una distribución conjunta de variables dependientes.

```
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
```

Conclusión:

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.