Módulo VI

Taller de IA y CD

Maciel-Cruz, EJ

2025-06-12

Contenidos

1 Estadística Descriptiva **2**
1.1 Visualización de la Base . . . 2
1.2 Pruebas de Ajuste . . . 7
1.2.1 Pruebas de Normalidad . . . 7
1.2.2 Pruebas de Homocedasticidad . . . 18
1.3 Pruebas de Comparación de Medias . . . 18
1.3.1 Prueba t de una Muestra . . . 18
1.3.2 Prueba de t de Muestras Independientes . . . 20
1.3.3 Prueba de t de Muestras Pareadas . . . 21
1.3.4 Prueba de ANOVA . . . 21
1.4 Prueba No Paramétricas . . . 22
1.4.1 Prueba de U de Mann-Whitney . . . 22
1.4.2 Prueba de Wilcoxon . . . 23
1.4.3 Prueba de Kruskal-Wallis . . . 23
1.4.4 Prueba de Friedman . . . 23
1.5 Pruebas de Proporciones . . . 24
1.5.1 Prueba de Chi-cuadrada (²) . . . 25
1.5.2 Prueba Exacta de Fisher . . . 26
1.5.3 Prueba Z de Dos Proporciones . . . 27
1.6 Correlación . . . 28
1.6.1 Correlación de Pearson . . . 28
1.6.2 Correlación de Spearman . . . 28
1.6.3 Correlación de Kendall . . . 29
1.7 Modelos de Regresión . . . 29
1.7.1 Regresión Lineal Simple . . . 30
1.7.2 Regresión Lineal Múltiple . . . 32
1.7.3 Regresión Logística Binomial . . . 33

1.8 Pruebas Post-Hoc 35
1.8.1 Prueba de Tukey 35
1.8.2 Prueba de Bonferroni 36
1.8.3 Prueba de Scheffé 36

1 Estadística Descriptiva

En esta sección se dará un breve repaso de los resúmenes que hemos hecho en el pre-procesado, pero se agregarán las funciones para extraer la estadística de interés, además de exportarla a un formato más manejable.

1.1 Visualización de la Base

Cargamos bases y librerías:

```
pacman::p_load(tidyverse,GGally,esquisse,ggThemeAssist,summarytools,readxl,viridis,hrbrtheme
BD2_R <- read_excel("BD2_R.xlsx")
BD2 <- BD2_R
```

En primer lugar, recordemos que debemos conocer cómo se encuentra constituida nuestra base de datos. Para ellos se hace lo siguiente:

```
BD_clases <- sapply(BD2, class)
info_cols <- data.frame(
  Columna = names(BD_clases),
  Clase = unname(BD_clases)
  )
print(info_cols)
```

```
          Columna     Clase
1              id character
2      colesterol   numeric
3         glucosa   numeric
4             hdl   numeric
5   col_hdl_ratio   numeric
6           hb1ac   numeric
7    diabetes_log   logical
8    lugar_origen character
9            edad   numeric
```

```
10           sexo character
11         altura   numeric
12           peso   numeric
13            imc   numeric
14 act_fisica_fac character
15          tx_ta   logical
16          tas_1   numeric
17          tad_1   numeric
18          tas_2   numeric
19          tad_2   numeric
20        cintura   numeric
21         cadera   numeric
```

Ahora recordamos el tipo de dato de cada variable.

Ahora demos un vistazo a las variables:

```
summary(BD2)
```

```
     id              colesterol       glucosa           hdl        
 Length:403         Min.   : 78.0   Min.   : 48.0   Min.   : 12.00  
 Class :character   1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00  
 Mode  :character   Median :204.0   Median : 89.0   Median : 46.00  
                    Mean   :207.8   Mean   :106.7   Mean   : 50.45  
                    3rd Qu.:230.0   3rd Qu.:106.0   3rd Qu.: 59.00  
                    Max.   :443.0   Max.   :385.0   Max.   :120.00  
                    NA's   :1                       NA's   :1       
 col_hdl_ratio        hb1ac       diabetes_log    lugar_origen      
 Min.   : 1.500   Min.   : 2.68   Mode :logical   Length:403        
 1st Qu.: 3.200   1st Qu.: 4.38   FALSE:325       Class :character  
 Median : 4.200   Median : 4.84   TRUE :65        Mode  :character  
 Mean   : 4.522   Mean   : 5.59   NA's :13                          
 3rd Qu.: 5.400   3rd Qu.: 5.60                                     
 Max.   :19.300   Max.   :16.11                                     
 NA's   :1        NA's   :13                                        
     edad           sexo               altura           peso       
 Min.   :19.00   Length:403         Min.   :1.321   Min.   :  0.00  
 1st Qu.:34.00   Class :character   1st Qu.:1.600   1st Qu.: 66.22  
 Median :45.00   Mode  :character   Median :1.676   Median : 80.29  
 Mean   :46.85                      Mean   :1.677   Mean   : 82.41  
 3rd Qu.:60.00                      3rd Qu.:1.753   3rd Qu.: 93.89  
 Max.   :92.00                      Max.   :1.930   Max.   :169.64  
```

```
                                NA's   :5
      imc         act_fisica_fac       tx_ta             tas_1
 Min.   : 0.00   Length:403         Mode :logical   Min.   : 90.0
 1st Qu.:23.89   Class :character   FALSE:141       1st Qu.:121.2
 Median :28.10   Mode  :character   TRUE :262       Median :136.0
 Mean   :29.39                                      Mean   :136.9
 3rd Qu.:33.88                                      3rd Qu.:146.8
 Max.   :64.20                                      Max.   :250.0
 NA's   :5                                          NA's   :5
     tad_1            tas_2           tad_2           cintura
 Min.   : 48.00   Min.   : 92.0   Min.   : 49.00   Min.   : 66.04
 1st Qu.: 75.00   1st Qu.:129.0   1st Qu.: 76.00   1st Qu.: 83.82
 Median : 82.00   Median :140.0   Median : 85.00   Median : 93.98
 Mean   : 83.32   Mean   :144.4   Mean   : 85.17   Mean   : 96.27
 3rd Qu.: 90.00   3rd Qu.:158.0   3rd Qu.: 94.00   3rd Qu.:104.14
 Max.   :124.00   Max.   :238.0   Max.   :129.00   Max.   :142.24
 NA's   :5        NA's   :6       NA's   :6        NA's   :2
     cadera
 Min.   : 76.20
 1st Qu.: 99.06
 Median :106.68
 Mean   :109.32
 3rd Qu.:116.84
 Max.   :162.56
 NA's   :2
```

Ya tenemos un mini resumen descriptivo.

Ahora veamos las variables categóricas:

```
BD2 [-1] %>%
  select_if(is.character) %>%
  sapply(unique)
```

```
$lugar_origen
[1] "Jal" "Mor" "Ags"

$sexo
[1] "mujer"  "hombre"

$act_fisica_fac
[1] "Bajo"       "Sedentario" "Alto"       NA
```

Recordemos que la primera variable es el id, por lo que no la deseamos ver. Usamos la función "select_if" para que si cumple con una condición (que es "is.character"), lleve a cabo la siguiente operación, que es "sapply", valores únicos.

> **Tip**
>
> También lo podemos usar para ver variables de tipo numérico, lógico, factor…

Ahora, usemos el resumen gráfico que habíamos trabajado:

```
view(dfSummary(BD2))
```

```
Switching method to 'browser'
```

```
Output file written: C:\Users\Hack\AppData\Local\Temp\RtmpGQMSH6\file2d54d314994.html
```

Bueno, ya tenemos un vistazo de nuestra base de datos, y ahora recordemos que podemos hacer resúmenes estadísticos con el mismo paquete de summarytools. Hagamos una estadística descriptiva por tipo de variable en dos objetos diferentes:

```
BD2_descr <- descr(BD2, transpose = T, stats = "common")
BD2_freq <- freq(BD2[-1])
BD2_descr
```

```
Non-numerical variable(s) ignored: id, diabetes_log, lugar_origen, sexo, act_fisica_fac, tx_
```

```
Descriptive Statistics
BD2
N: 403
```

	Mean	Std.Dev	Min	Median	Max	N.Valid	N	Pct.Val
altura	1.68	0.10	1.32	1.68	1.93	398.00	403.00	98
cadera	109.32	14.37	76.20	106.68	162.56	401.00	403.00	99
cintura	96.27	14.55	66.04	93.98	142.24	401.00	403.00	99
col_hdl_ratio	4.52	1.73	1.50	4.20	19.30	402.00	403.00	99
colesterol	207.85	44.45	78.00	204.00	443.00	402.00	403.00	99
edad	46.85	16.31	19.00	45.00	92.00	403.00	403.00	100
glucosa	106.67	53.08	48.00	89.00	385.00	403.00	403.00	100
hb1ac	5.59	2.24	2.68	4.84	16.11	390.00	403.00	96
hdl	50.45	17.26	12.00	46.00	120.00	402.00	403.00	99

imc	29.39	7.84	0.00	28.10	64.20	398.00	403.00	98
peso	82.41	22.35	0.00	80.29	169.64	403.00	403.00	100
tad_1	83.32	13.59	48.00	82.00	124.00	398.00	403.00	98
tad_2	85.17	12.64	49.00	85.00	129.00	397.00	403.00	98
tas_1	136.90	22.74	90.00	136.00	250.00	398.00	403.00	98
tas_2	144.40	21.15	92.00	140.00	238.00	397.00	403.00	98

En la función de "stats" podemos seleccionar "common", la cual contiene las estadísticas habitualmente utilizadas.

Naturalmente, queremos guardarlo para ponerlo en nuestros informes, esto se logra así:

```
capture.output (BD2_descr, file = "BD2_numéricas.txt")
```

```
Non-numerical variable(s) ignored: id, diabetes_log, lugar_origen, sexo, act_fisica_fac, tx_t
```

```
capture.output (BD2_freq, file = "BD2_categóricas.txt")
```

```
Variable(s) ignored: colesterol, glucosa, hdl, col_hdl_ratio, hb1ac, edad, peso, imc
```

Podemos guardar nuestros resúmenes ya sea en formato .txt o word (.doc), solo hay que cambiar la extensión. La función de "capture.output" nos permite guardar objetos que tienen contenido de tipo de texto.

Ya tenemos estadística descriptiva básica y el método para exportarlo a nuestra computadora. Ahora, realizaremos en orden las pruebas habituales que se hacen previo a la pruebas estadísticas normales.

Hagamos ahora dos ejercicios:

> Hagan un dfsummary con la base completa.
>
> Hagan un ggpairs con la base completa (quiten el id).

Pueden utilizar los códigos que hemos trabajado en días previos.

> **Note**
>
> Para fines del curso, solamente se mostrará el cómo realizar en R cada prueba mencionada sin considerar si es la más adecuada o no por el tipo de dato.

1.2 Pruebas de Ajuste

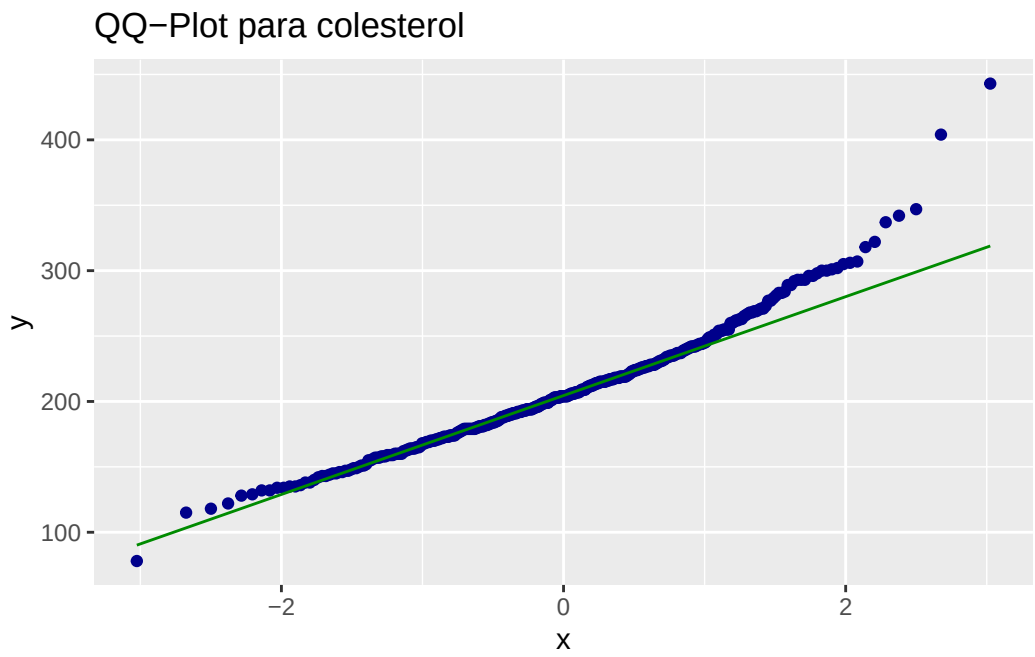
1.2.1 Pruebas de Normalidad

Las pruebas de normalidad se realizan para evaluar si una variable numérica presenta una distribución paramétrica. Previo a la pruebas como tal, se puede evaluar mediante un histograma o un QQ-plot. Como ya contamos el conocimiento base del histograma, pasemos al QQ-plot.

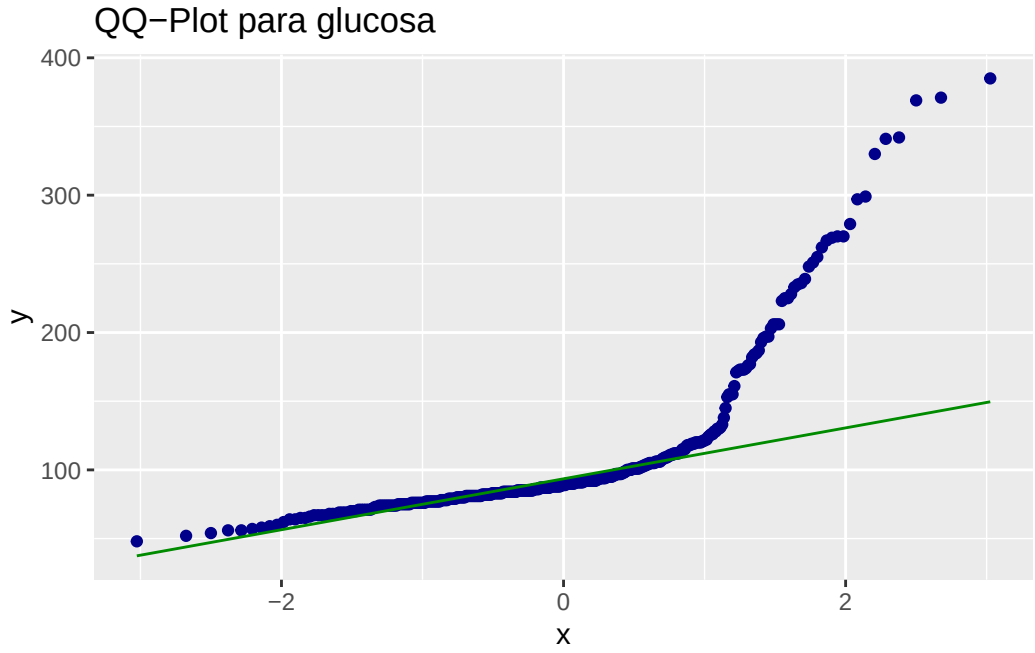
```
variables <- BD2[-1] %>%
  select_if(
    is.numeric
    ) %>%
  colnames()

plots <- lapply(variables,
                function(var) {
  ggplot(BD2,
         aes(sample = .data[[var]]
             )
         ) +
    stat_qq(color = "blue4") +
    stat_qq_line(color = "green4") +
    ggtitle(paste("QQ-Plot para", var))
})
plots
```

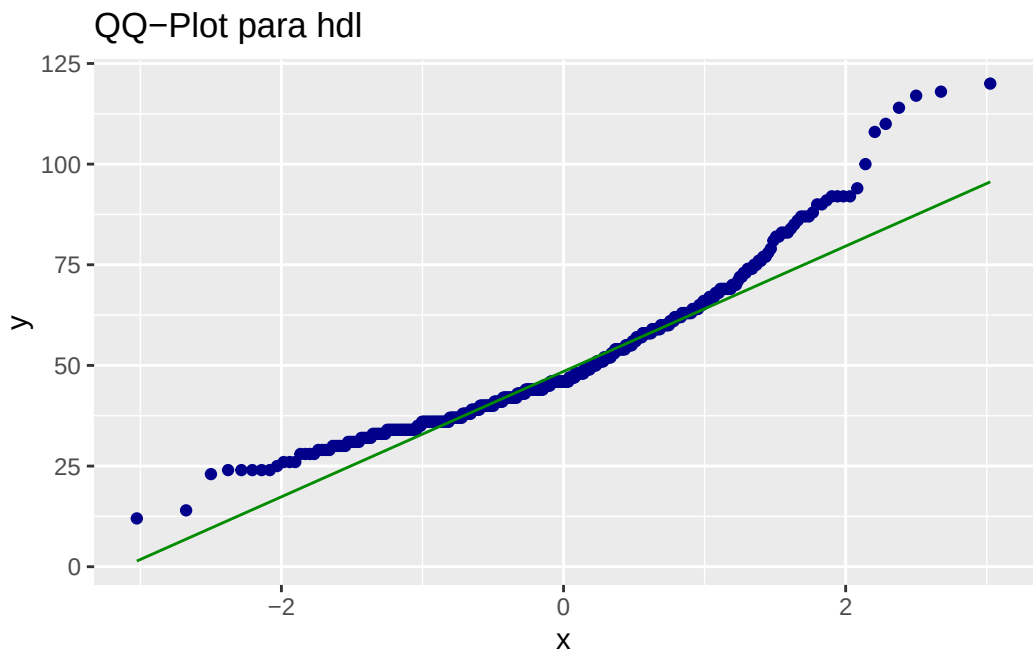
```
[[1]]
```


QQ−Plot para colesterol
400
300
200
100
y
−2
0
2
x

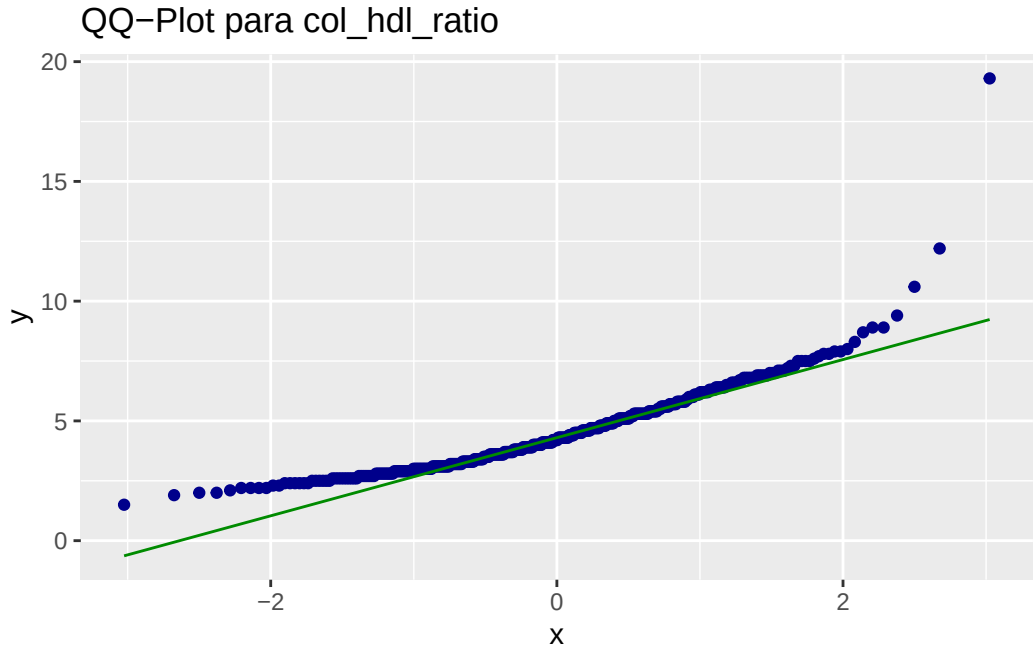
```
[[2]]
```


QQ−Plot para glucosa
400
300
200
100
y
−2
0
2
x

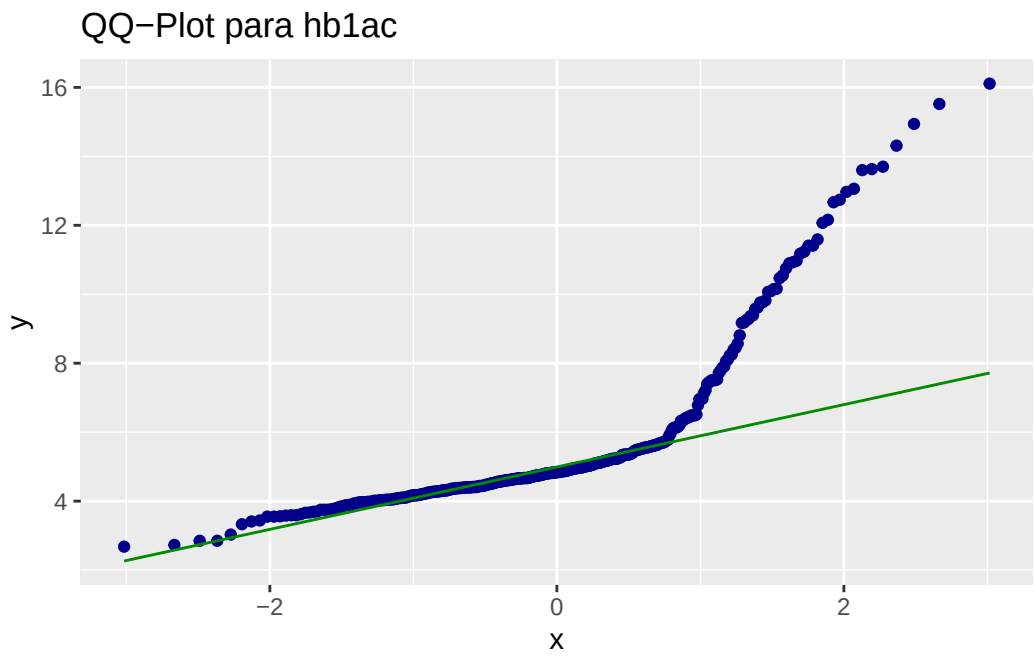
```
[[3]]
```


QQ−Plot para hdl
125
100
75
50
25
0
y
−2
0
2
x

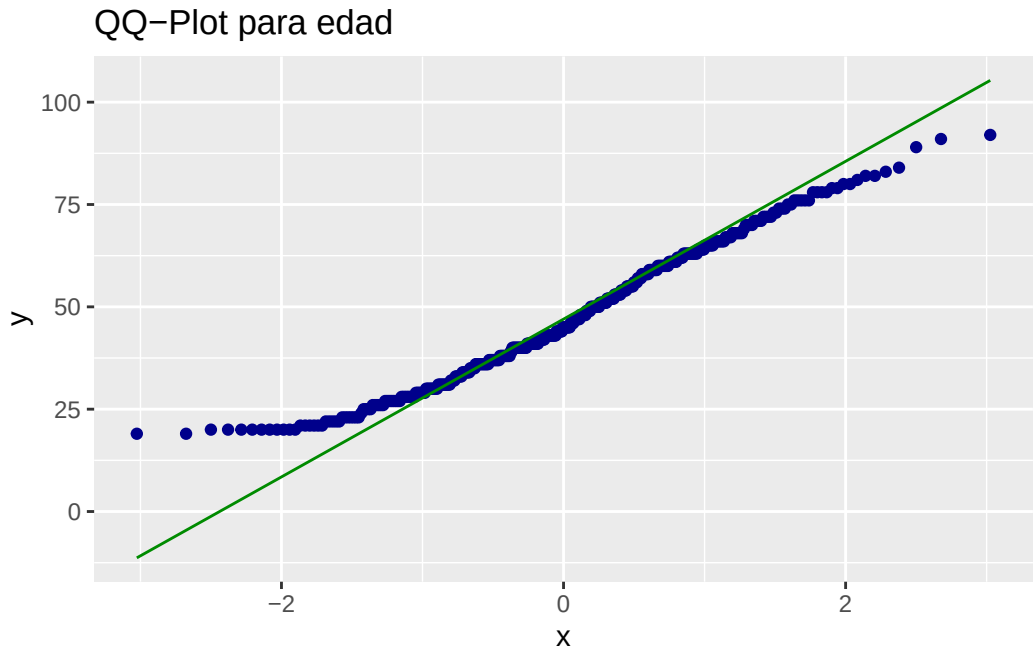
```
[[4]]
```


QQ−Plot para col_hdl_ratio
20
15
10
5
0
y
−2
0
2
x

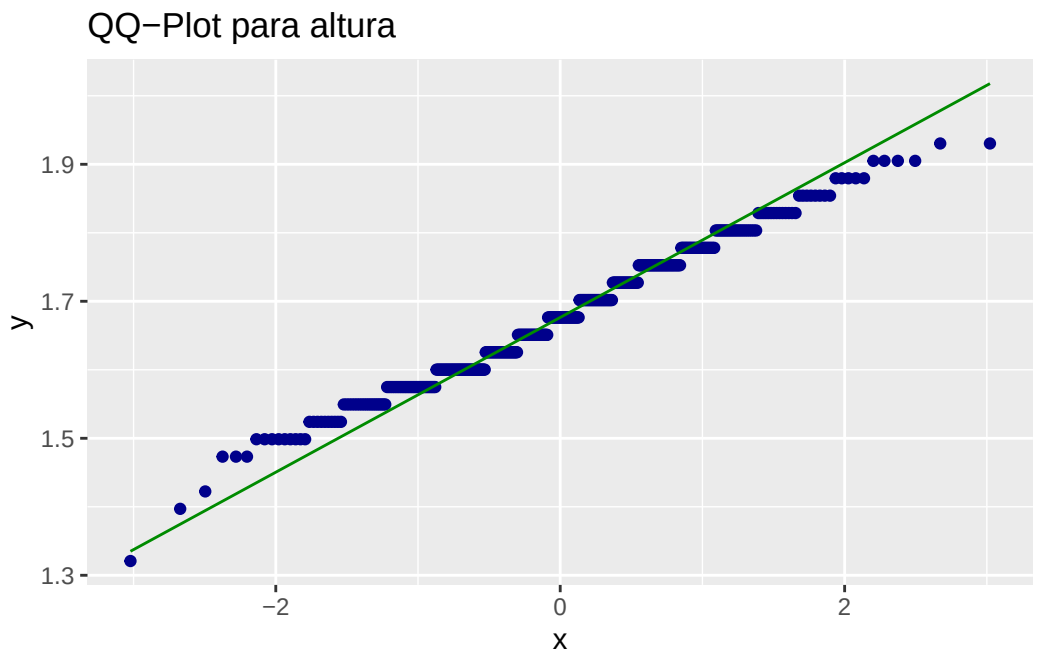
```
[[5]]
```


QQ−Plot para hb1ac
16
12
8
4
y
−2
0
2
x

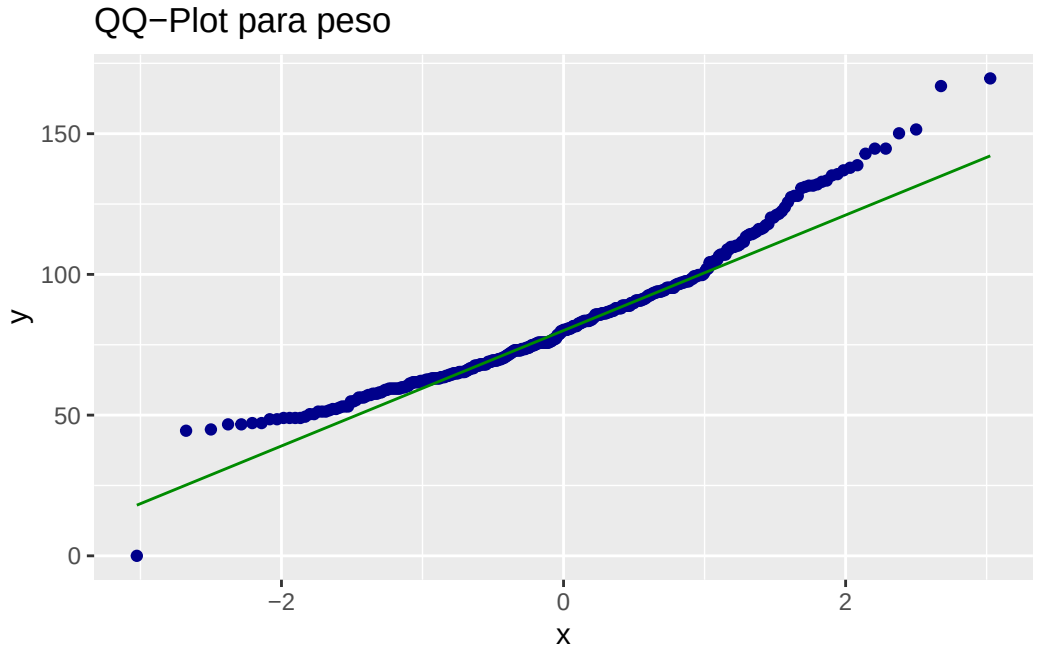
```
[[6]]
```


QQ−Plot para edad
100
75
50
25
0
y
−2
0
2
x

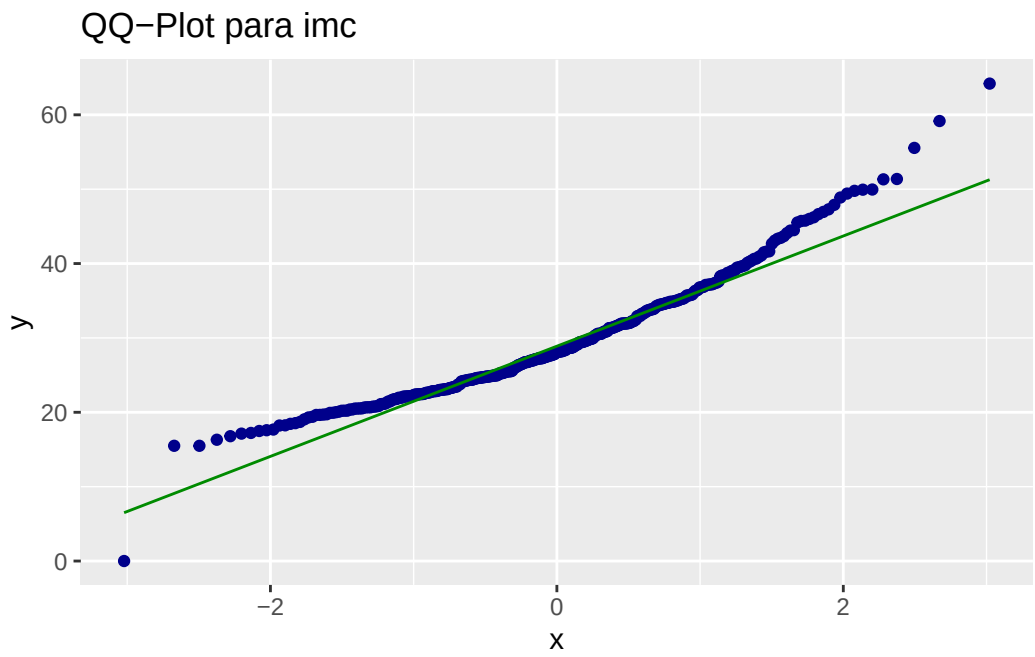
```
[[7]]
```


QQ−Plot para altura
1.9
1.7
1.5
1.3
y
−2
0
2
x

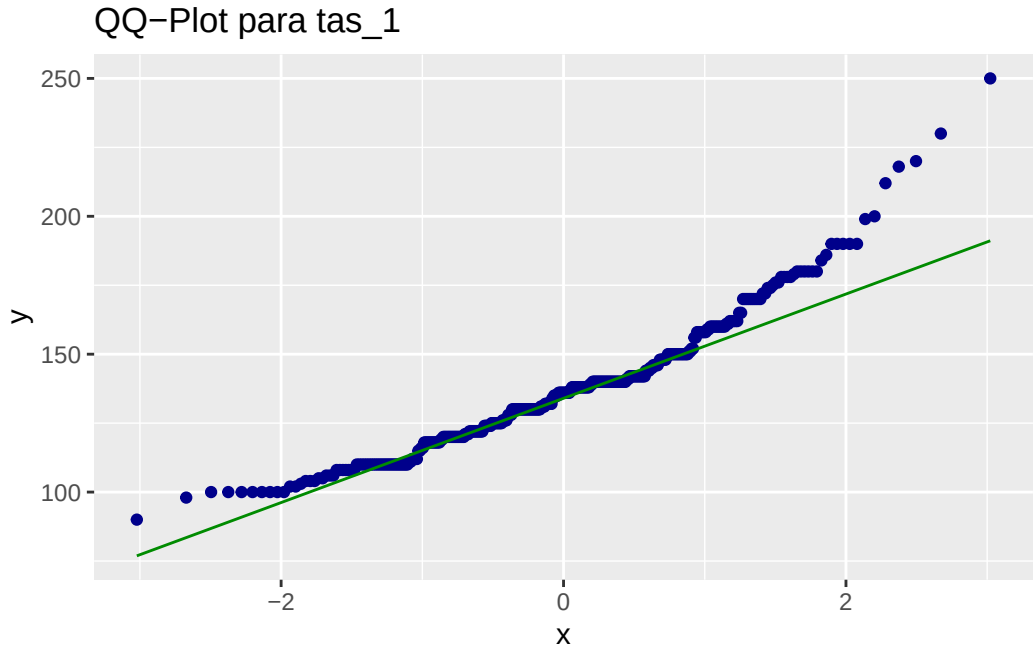
```
[[8]]
```


QQ−Plot para peso
150
100
50
0
y
−2
0
2
x

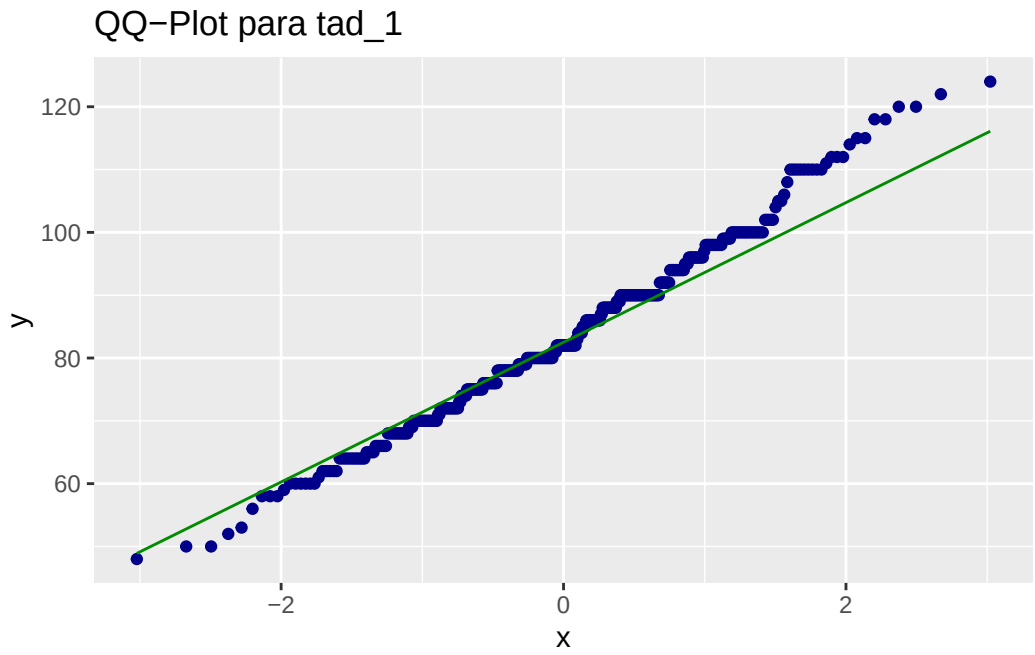
```
[[9]]
```


QQ−Plot para imc
60
40
20
0
y
−2
0
2
x

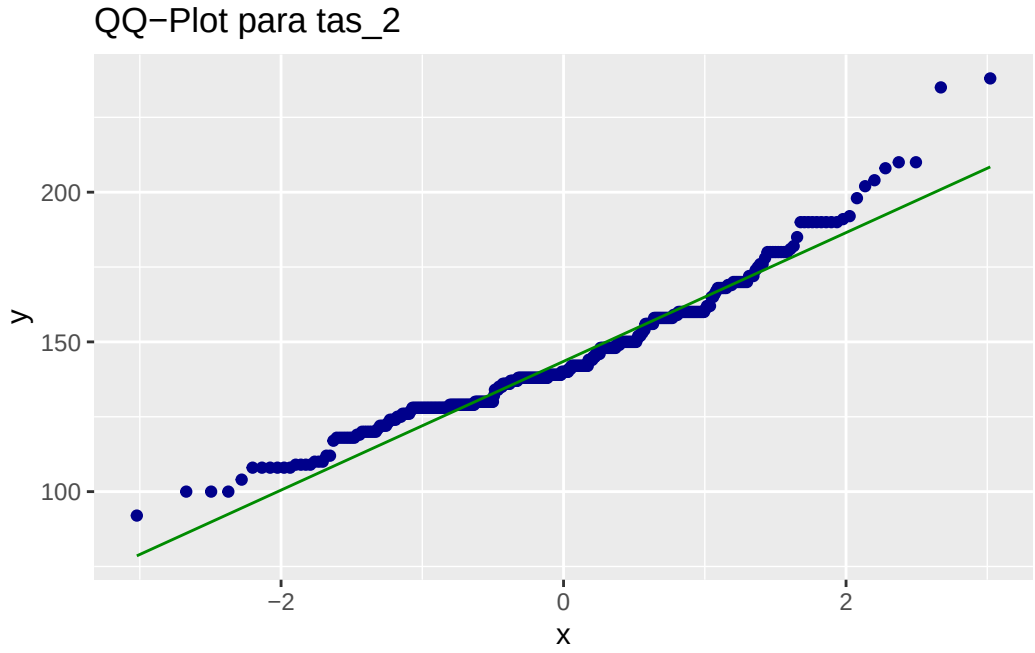
```
[[10]]
```


QQ−Plot para tas_1
250
200
150
100
y
−2
0
2
x

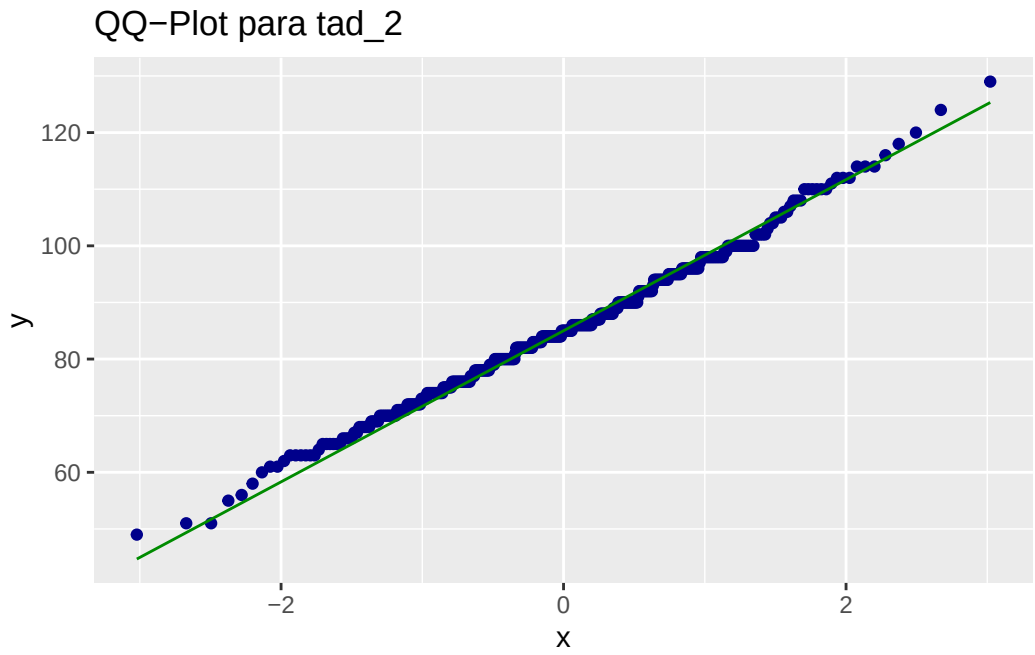
```
[[11]]
```


QQ−Plot para tad_1
120
100
80
60
y
−2
0
2
x

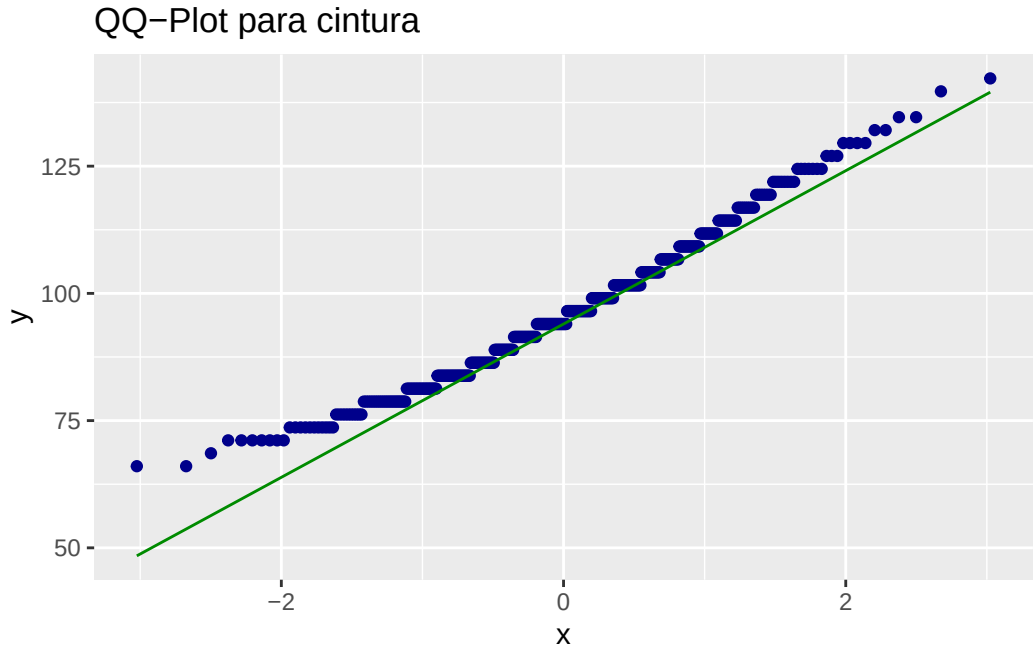
```
[[12]]
```


QQ−Plot para tas_2
200
150
100
y
−2
0
2
x

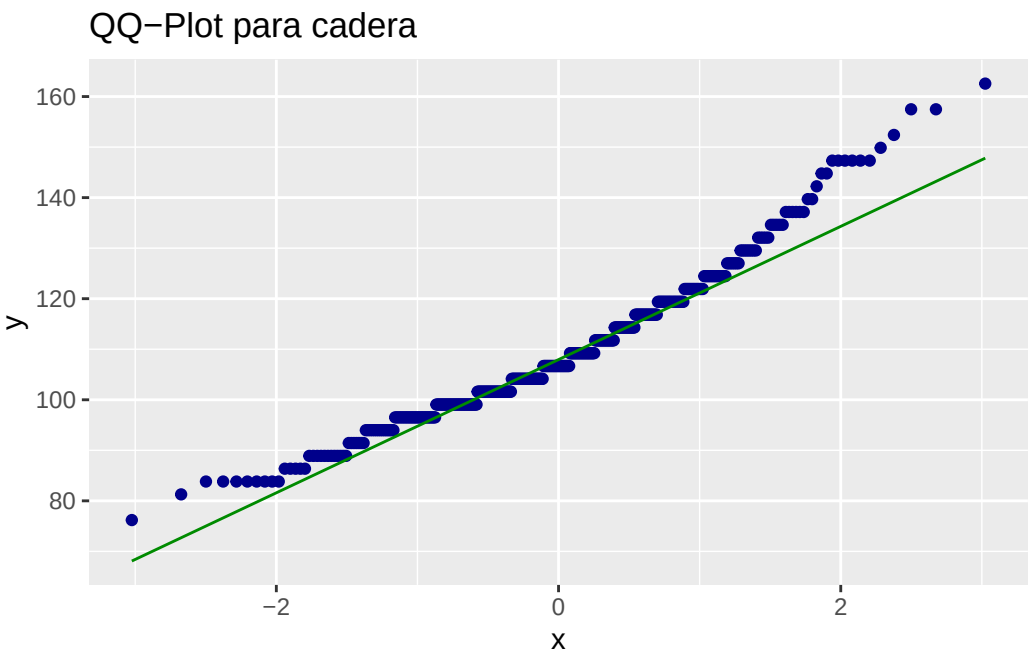
```
[[13]]
```


QQ−Plot para tad_2
120
100
80
60
y
−2
0
2
x

```
[[14]]
```


QQ−Plot para cintura
125
100
75
50
y
−2
0
2
x

```
[[15]]
```



En el código anterior primero generamos un objeto llamado “variables” que contenga un vector con los nombres de las variables que son de tipo numérico.

En la segunda sección, primero creamos un objeto que se va a llamar “plots”. Luego hacemos una “lapply” (list apply) que utiliza el objeto “variables” y se le va a aplicar una función con un nombre genérico, que es “var” (puede ser el que tú quieras). Llamamos la librería “ggplot” de forma habitual, y para el aes usamos una función “sample = .data[[var]]” que es similar a usar el símbolo de “$” para llamar una columna, pero como queremos automatizar todo a través del vector, usamos este método. Hacemos llamado para la función de la distribución de nuestros datos en puntos y la distribución normal con línea con “stat_qq” y “stat_qq_line”, respectivamente. Finalmente, usamos “ggtitle(paste(”QQ-Plot para”, var” para hacer títulos genéricos que empiezan con “QQ-Plot para” e inmediatamente se pega el nombre de la variable tal como viene en nuestro vector.

Naturalmente, es un código un poco extenso, pero lo puedes reutilizar y solamente pegas el nombre de tu base de datos.

Ahora, procedemos a la prueba de normalidad, asumamos que queremos realizar la prueba de Shapiro-Wilk:

```
shapiro.test(BD2$colesterol)
```

```
    Shapiro-Wilk normality test
```

```
data:  BD2$colesterol
W = 0.95939, p-value = 4.296e-09
```

Naturalmente, podemos realizar la prueba de Kolmogorv-Smirnov, hay que considerar que como existen valores repetidos, nos va a arrojar un error:

```
ks.test(BD2$colesterol, 'pnorm')
```

```
Warning in ks.test.default(BD2$colesterol, "pnorm"): ties should not be present
for the one-sample Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  BD2$colesterol
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided
```

También podemos llevar a cabo la prueba de Anderson-Darling, la cual se puede ajustar a nuestra variable:

```
ad.test(BD2$colesterol)
```

```
    Anderson-Darling normality test

data:  BD2$colesterol
A = 2.7829, p-value = 5.149e-07
```

Por supuesto, podemos usar la función "sapply" para generar las pruebas de normalidad a todo:

```
BD2 %>%
  select_if(is.numeric) %>%
  sapply(shapiro.test)
```

```
          colesterol                    glucosa
statistic 0.9593867                     0.6476578
p.value   4.295804e-09                  5.454664e-28
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
```

```
          hdl                           col_hdl_ratio
statistic 0.9235699                     0.8696703
p.value   1.917537e-13                  7.226449e-18
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
          hb1ac                         edad
statistic 0.7218346                     0.9755724
p.value   5.512283e-25                  2.653284e-06
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
          altura                        peso
statistic 0.98794                       0.9533511
p.value   0.002227658                   5.569332e-10
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
          imc                           tas_1
statistic 0.9502796                     0.9376639
p.value   2.586659e-10                  7.306905e-12
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
          tad_1                         tas_2
statistic 0.9904326                     0.9523933
p.value   0.01091829                    5.161281e-10
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
          tad_2                         cintura
statistic 0.9943093                     0.9788179
p.value   0.1464415                     1.326393e-05
method    "Shapiro-Wilk normality test" "Shapiro-Wilk normality test"
data.name "X[[i]]"                      "X[[i]]"
          cadera
statistic 0.9592809
p.value   4.290232e-09
method    "Shapiro-Wilk normality test"
data.name "X[[i]]"
```

> **Tip**
>
> Puedes cambiar la prueba de normalidad que desees dentro de la función de “sapply”.

1.2.2 Pruebas de Homocedasticidad

Las pruebas de homocedasticidad describen si diferentes grupos presentan varianzas iguales, la prueba de Levene para distribuciones no paramétricas se lleva a cabo de la siguiente manera:

```
leveneTest(hdl ~ sexo, data = BD2)
```

```
Warning in leveneTest.default(y = y, group = group, ...): group coerced to
factor.
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value Pr(>F)
group   1  0.7902 0.3746
      400
```

La sintaxis es llamar la función "leveneTest", primero la variable numérica, luego los factores o grupos, luego la base de datos de la que provienen. La naturaleza de la función no acepta operadores de pipa, por lo que debemos de realizar cada prueba por separado.

Para la prueba de Bartlett con enfoque a distribuciones paramétricas, se hace de la siguiente manera:

```
bartlett.test(hdl ~ sexo, data = BD2)
```

```
    Bartlett test of homogeneity of variances

data:  hdl by sexo
Bartlett's K-squared = 0.026622, df = 1, p-value = 0.8704
```

Notarás que es exactamente igual... Solo hay que cambiar el nombre de la función.

1.3 Pruebas de Comparación de Medias

1.3.1 Prueba t de una Muestra

Esta prueba compara la media de de una variable con un valor predeterminado, conocido o hipotético, se realiza de la siguiente manera:

```r
t.test(BD2$glucosa, mu = 100)
```

```
    One Sample t-test

data:  BD2$glucosa
t = 2.5237, df = 402, p-value = 0.012
alternative hypothesis: true mean is not equal to 100
95 percent confidence interval:
 101.4748 111.8701
sample estimates:
mean of x
 106.6725
```

Como ya vimos, podemos agrupar nuestros datos a como lo necesitemos, y luego hacer la prueba estadística. Asumamos que queremos hacer la prueba de una muestra para dos subgrupos de la variable "diabetes_log":

```r
BD2 %>%
  group_by(diabetes_log) %>%
  summarise(
    valor_p = t.test(glucosa, mu = 100)$p.value,
    media = mean(glucosa, na.rm = TRUE),
    n = n()
  )
```

```
# A tibble: 3 x 4
  diabetes_log  valor_p media     n
  <lgl>           <dbl> <dbl> <int>
1 FALSE        1.48e-10  90.9   325
2 TRUE         3.12e-13 190.     65
3 NA           5.35e- 4  86.7    13
```

En este código sucede lo siguiente: - "valor_p = t.test(glucosa, mu = 100)$p.value": creamos el objeto de "valor_p" que contiene solamente el resultado del valor de p de la prueba t de Student contra la media de 100. - "media = mean(glucosa, na.rm = TRUE)": guardamos la media de la glucosa por grupo de nuestra base de datos. - "n = n()": guardamos la n u observaciones.

Listo, nos entrega una cuadro muy visual en nuestra terminal.

1.3.2 Prueba de t de Muestras Independientes

Esta prueba estadística compara las medias de dos grupos que no están relacionados o pareados, se realiza de la siguiente manera:

```
t.test(glucosa ~ diabetes_log, data = BD2_R, var.equal = T)
```

```
    Two Sample t-test

data:  glucosa by diabetes_log
t = -18.499, df = 388, p-value < 2.2e-16
alternative hypothesis: true difference in means between group FALSE and group TRUE is not e
95 percent confidence interval:
 -109.18499  -88.20578
sample estimates:
mean in group FALSE  mean in group TRUE
           90.88923           189.58462
```

```
t.test(glucosa ~ diabetes_log, data = BD2_R)
```

```
    Welch Two Sample t-test

data:  glucosa by diabetes_log
t = -9.9809, df = 66.547, p-value = 7.455e-15
alternative hypothesis: true difference in means between group FALSE and group TRUE is not e
95 percent confidence interval:
 -118.43529  -78.95548
sample estimates:
mean in group FALSE  mean in group TRUE
           90.88923           189.58462
```

La primera línea hace la prueba t de Student, asumiendo las varianzas iguales, mientras que la segunda hace el test de Welch para varianzas no iguales.

La sintaxis es la siguiente: primero, llamamos a la función "t.test", seleccionamos la variables numérica de la base de datos, usamos el símbolo "~", seleccionamos la variable agrupadora, y seleccionamos si hay igualdad de varianzas (T para igualdad, F para no igualdad).

1.3.3 Prueba de t de Muestras Pareadas

En esta prueba analizamos la diferencia de medias entre dos grupos que tengan datos pre y post, o que estén pareados (antes o después de un manejo, tratamiento, prueba, etc.).

Se hace la siguiente manera:

```
BD2 %>%
  group_by(tx_ta) %>%
  summarise(
    dif_medias = mean(tas_1 - tas_2, na.rm = T),
    valor_p    = t.test(tas_1, tas_2, paired = T)$p.value,
    n           = n()
  )
```

```
# A tibble: 2 x 4
  tx_ta dif_medias  valor_p     n
  <lgl>      <dbl>    <dbl> <int>
1 FALSE       2.38 1.09e- 2   141
2 TRUE      -13.4  2.42e-32   262
```

Ya que tenemos la variable de aquellos tomaron un manejo para TA, los podemos agrupar por esa variable, y luego llevamos a cabo la misma prueba que hicimos, pero anexamos el argumento "paired = T".

1.3.4 Prueba de ANOVA

La prueba de ANOVA (Análisis de Varianza), básicamente es una extensión de la prueba de t que admite más de dos grupos. Se utiliza cuando en nuestra variable agrupadora tenemos tres o más niveles.

Se hace de la siguiente manera:

```
ANOVA_test <- aov(BD2$colesterol ~ BD2$act_fisica_fac)
summary(ANOVA_test)
```

```
                   Df Sum Sq Mean Sq F value Pr(>F)
BD2$act_fisica_fac   2  16999    8500   4.504 0.0116 *
Residuals          387 730265    1887
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
13 observations deleted due to missingness
```

En el caso de ANOVA se debe crear el objeto y después se hace el resumen de los datos para obtener el valor de significancia. La sintaxis es igual que en el resto de las pruebas que hemos visto. Solo nos indica si hay diferencia en algún grupo, para conocer cuál debemos realizar pruebas post hoc.

Aquí no cambies el nombre del objeto "ANOVA_test", ya que lo usaremos más adelante.

1.4 Prueba No Paramétricas

Entre las pruebas no paramétricas se puede seleccionar entre los siguientes supuestos:

- Dos grupos independientes: U de Mann-Whitney
  - TA entre un grupo con manejo y otro grupo sin manejo
- Dos grupos dependientes o relacionados: Wilcoxon
  - TA pre y post manejo en los mismos individuos
- Tres o más grupos independientes: Kruskal-Wallis
  - Glucosas entre un grupo con manejo farmacológico, otro con solo dieta, y otro con ambos manejos
- Tres o más grupos dependientes o relacionados: Friedman
  - Glucosas pre y post manejos: grupo con manejo farmacológico, grupo con manejo de dieta, grupo con manejo de dieta y farmacológico

1.4.1 Prueba de U de Mann-Whitney

Es exactamente igual que la prueba t.

```
wilcox.test(tas_1 ~ tx_ta, data = BD2_R)
```

```
    Wilcoxon rank sum test with continuity correction

data:  tas_1 by tx_ta
W = 32038, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Notemos que las pruebas pueden detectar niveles en formato lógico, por lo que se separa en los dos grupos y nos arroja el valor de significancia calculado.

1.4.2 Prueba de Wilcoxon

La prueba de Wilcoxon es la alternativa no paramétrica para la prueba t de muestras pareadas, se realiza de la siguiente manera:

```
wilcox.test(BD2$tas_1, BD2$tas_2, paired = T)
```

```
    Wilcoxon signed rank test with continuity correction

data:  BD2$tas_1 and BD2$tas_2
V = 15280, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

Exactamente igual que "t.test".

1.4.3 Prueba de Kruskal-Wallis

La prueba de Kruskal-Wallis es la alternativa no paramétrica para la prueba de ANOVA de grupos no pareados o independientes, se realiza de la siguiente manera:

```
kruskal.test(BD2$colesterol~BD2$lugar_origen)
```

```
    Kruskal-Wallis rank sum test

data:  BD2$colesterol by BD2$lugar_origen
Kruskal-Wallis chi-squared = 2.3246, df = 2, p-value = 0.3128
```

Y así, con cualquier variable que deseemos.

1.4.4 Prueba de Friedman

Es la alternativa no paramétrica de la prueba de ANOVA para grupos pareados. Aquí evaluamos tres o más grupos consecuentes o dependientes. Para ello, vamos a crear una pequeña base de datos:

```
datos_F <- data.frame(sujeto = rep(1:5, each=4),
                      medicamento = rep(c("A", "B", "C", "D"), times=5),
                      biomarcador = c(30, 28, 16, 34, 14, 18, 10, 22, 24, 20,
                             18, 30, 38, 34, 20, 44, 26, 28, 14, 30))
summary(datos_F)
```

```
     sujeto  medicamento         biomarcador
 Min.   :1   Length:20          Min.   :10.0
 1st Qu.:2   Class :character   1st Qu.:18.0
 Median :3   Mode  :character   Median :25.0
 Mean   :3                      Mean   :24.9
 3rd Qu.:4                      3rd Qu.:30.0
 Max.   :5                      Max.   :44.0
```

Ahora, podemos proceder a la prueba:

```
friedman.test(y=datos_F$biomarcador,
              groups=datos_F$medicamento,
              blocks=datos_F$sujeto)
```

```
    Friedman rank sum test

data:  datos_F$biomarcador, datos_F$medicamento and datos_F$sujeto
Friedman chi-squared = 13.56, df = 3, p-value = 0.00357
```

La dificultad radica en la construcción de la base de datos. En primera instancia debe contener tres variables: la variable de datos a analizar ("y"), la variable de los niveles a agrupar ("groups") y la variable de individuos o sujetos de estudio ("blocks").

Naturalmente, los datos deben de estar equilibrados, debe de existir el mismo número de observaciones para cada sujeto y grupo, en este caso 4 valores por biomarcador por grupo y por sujeto, ya que cada sujeto tiene una medición del biomarcador por fármaco (A, B, C, D).

1.5 Pruebas de Proporciones

Las pruebas de proporción se dividen en tres:

- Comparación de frecuencias esperadas vs observadas: Chi-cuadrada (²)
- Comparación de frecuencias esperadas vs observadas si en una observación existe un valor menor de cinco o si el tamaño total de la muestra es menor de 20 o si hay una despropoción muy grande entre filas o columnas: Exacta de Fisher
- Comparación de proporciones entre dos grupos: Prueba Z

1.5.1 Prueba de Chi-cuadrada (^{2})

Lo primero que debemos de hacer es analizar las proporciones por grupos de las variables que queremos comparar, primero se hacen las tablas de contingencia:

```
table(
  BD2$sexo,
  BD2$diabetes_log
  )
```

```
        FALSE TRUE
  hombre   133   29
  mujer    192   36
```

Ya vemos la distribución de los datos que es proporcionado, no hay valores menores de cinco y la muestra es mayor de veinte.

```
chisq.test(
  BD2$sexo,
  BD2$diabetes_log
  )
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  BD2$sexo and BD2$diabetes_log
X-squared = 0.17105, df = 1, p-value = 0.6792
```

Y listo, no hay diferencia, así de simple es hacer las chi-cuadradas en R.

Podemos hacer la misma prueba con varios grupos, aquí no hay modificaciones:

```
table(
  BD2$diabetes_log,
  BD2$act_fisica_fac
  )
```

```
       Alto Bajo Sedentario
  FALSE   93  150         73
  TRUE     9   28         26
```

```r
chisq.test(
  BD2$diabetes_log,
  BD2$act_fisica_fac
  )
```

```
    Pearson's Chi-squared test

data:  BD2$diabetes_log and BD2$act_fisica_fac
X-squared = 11.217, df = 2, p-value = 0.003667
```

Y listo, podemos utilizar variables más de dos categorías sin problema.

1.5.2 Prueba Exacta de Fisher

Si se fallara en algún supuesto de los mencionados, procedemos a hacer exacta de Fisher. Sabemos de antemano que para el ejemplo previo lo más recomendado es la chi, pero por cuestiones de la base usaremos las mismas variables:

```r
table(
  BD2$diabetes_log,
  BD2$act_fisica_fac
  )
```

```
       Alto Bajo Sedentario
  FALSE   93  150         73
  TRUE     9   28         26
```

```r
fisher.test(
  BD2$diabetes_log,
  BD2$act_fisica_fac
  )
```

```
    Fisher's Exact Test for Count Data

data:  BD2$diabetes_log and BD2$act_fisica_fac
p-value = 0.004095
alternative hypothesis: two.sided
```

Y listo, misma sintaxis, diferente función

1.5.3 Prueba Z de Dos Proporciones

Si quisiéramos comparar dos proporciones, primeramente debemos obtener las frecuencias en una tabla cruzada:

```
addmargins(table(BD2$diabetes_log,
                 BD2$sexo))
```

```
       hombre mujer Sum
  FALSE    133   192 325
  TRUE      29    36  65
  Sum      162   228 390
```

La función "addmargins" nos genera la suma de totales por cada fila y columna. La función "table" nos genera una tabla cruzada de las variables que solicitemos.

Con las frecuencias por grupo, fila y columna, podemos preguntarnos si las frecuencias de mujeres son iguales o diferentes dependiendo si tienen o no diabetes. Podemos llevarlo a cabo de la siguiente manera:

```
prop.test(x = c(192,36),
          n = c(325,65))
```

```
    2-sample test for equality of proportions with continuity correction

data:  c(192, 36) out of c(325, 65)
X-squared = 0.17105, df = 1, p-value = 0.6792
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.1044480  0.1782942
sample estimates:
   prop 1    prop 2
0.5907692 0.5538462
```

Y no, las frecuencias no presentan una diferencia estadísticamente significativa. Luego podemos hacer la comparación con las variables que deseemos.

1.6 Correlación

Los modelos de correlación permiten evaluar una relación entre dos variables. Si observamos una tendencia lineal, en el eje de las "x" significa que no hay relación, si observamos un aumento o disminución progresiva o puntual, significa que existe relación en algún punto, pero se torna lineal o inversa si nos extendemos en alguno de los ejes.

Veremos tres tipos de correlación:

- Pearson: opción paramétrica para dos variables numéricas.
- Spearman: opción no paramétrica, se puede emplear una variable ordinal.
- Kendall: opción para dos variables ordinales.

1.6.1 Correlación de Pearson

Bastante simple de llevar a cabo, pero debemos recordar que no podemos tener datos NA:

```
cor(
  BD2$edad,
  BD2$colesterol,
  use = "complete.obs",
  method = "pearson"
  )
```

```
[1] 0.2331191
```

Recordemos que este es el método para calcular el grado de correlación.

1.6.2 Correlación de Spearman

Igual de simple de llevar a cabo:

```
cor(
  BD2$diabetes_log,
  BD2$glucosa,
  use = "complete.obs",
  method = "spearman"
  )
```

```
[1] 0.5325384
```

> **Tip**
>
> Recordemos que debemos tener valores numéricos para las correlaciones, por lo que tendríamos que llevar a cabo la transformación de variables.

1.6.3 Correlación de Kendall

Es correcto, es muy simple de llevar a cabo. El único inconveniente es que los niveles deben ser numéricos, o al menos lógicos. Además, recordemos que deben ser ordinales:

```
BD3 <- BD2 %>%
  drop_na() %>%
  mutate(
    act_fisica_fac = case_when(
      act_fisica_fac == "Sedentario" ~ 0,
      act_fisica_fac == "Bajo"       ~ 1,
      act_fisica_fac == "Alto"       ~ 2,
    )
  )

cor(
  BD3$act_fisica_fac,
  BD3$hdl,
  method = "kendall"
  )
```

```
[1] 0.1922452
```

En este caso, tenemos que crear una nueva base que primeramente no tenga datos omitidos. Luego transformamos la variable de interés, "act_fisica_fac" con los valores de 0 para la base, que es "Sedentario" y vamos adicionando los niveles. Para esto, hacemos uso de la función "case_when", que nos permite hacer una acción sí se encuentra determinada característica. Finalmente, corremos el modelo de correlación con el método de "kendall".

1.7 Modelos de Regresión

Los modelos de regresión nos permiten identificar la relación entre dos o más variables, cuál o cuáles tienen mayor relevancia en una variable de interés u objetivo o dependiente, además

de estimar los incrementos esperados para la variable dependiente acorde a la o las independientes (es decir, que tanto debería incrementarse o disminuirse la variable dependiente, si incrementamos la o las independientes).

Por fines de tiempo, solo se darán los primeros pasos

Veremos cuatro tipos de regresión:

- Regresión Lineal Simple: relación entre una variable dependiente continua y una variable independiente continua.
- Regresión Lineal Múltiple: relación entre una variable dependiente continua y múltiples variables independientes
- Regresión Logística: relación entre una variable dependiente binaria y una o más variables independientes.
- Regresión Logística Multinomial: relación entre una variable dependiente multifactorial y una o más variables independientes.

> **Note**
>
> El modelo de Regresión Logística Multinomial se llevará a cabo con el archivo "BD_ML.qmd", que se envió por aparte.

1.7.1 Regresión Lineal Simple

Para este tipo de modelo, vamos a crear unas variables de interés mediante código:

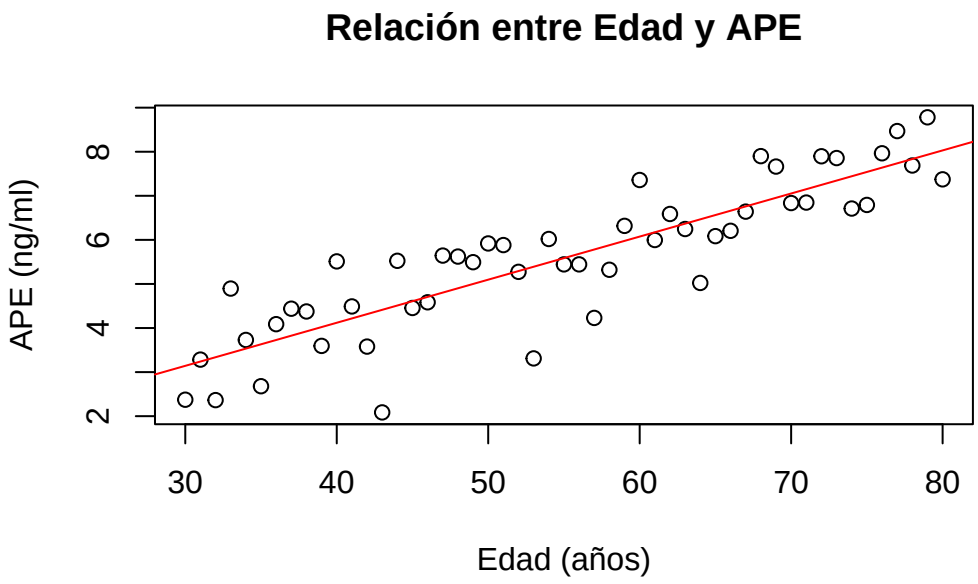
```
set.seed(1)
edad <- 30:80
APE <- .1 * edad + rnorm(50, mean = 0, sd = 1)
```

```
Warning in 0.1 * edad + rnorm(50, mean = 0, sd = 1): longer object length is
not a multiple of shorter object length
```

El código se interpreta así: "set.seed" es una semilla de aleatorización, se agrega un número para que los valores aleatorios que vayamos a generar sean tan similares que se pueda reproducir el resultado. Creamos la variable edad, que va de 30 a 80, asumamos que son años. Creamos la variable APE (Antígeno Prostático Específico), que es una variable aleatoria y calculada; multiplicamos .1 por la edad, y sumamos una distribución normal que incluya 50 datos, con una media de cero y una desviación estándar de 1. Lo anterior genera 50 datos aleatorios que tienen cierta relación con la edad, pero hay variabilidad.

El siguiente paso es crear un gráfico para evaluar si nuestros datos siguen una distribución lineal:

```r
plot(edad, APE, main = "Relación entre Edad y APE",
     xlab = "Edad (años)", ylab = "APE (ng/ml)")
abline(lm(APE ~ edad), col = "red")
```



La función de "abline" agrega una línea a través de los puntos y queremos que esté basada en un modelo lineal ("lm") ajustado a nuestras variables de interés.

Ahora, tenemos que crear un objeto que contenga nuestro modelo y solicitamos un resumen:

```r
RLS <- lm(APE ~ edad)
summary(RLS)
```

```
Call:
lm(formula = APE ~ edad)

Residuals:
     Min       1Q   Median       3Q      Max
-2.32775 -0.46734 -0.01834  0.63033  1.45985

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.209287   0.453374   0.462    0.646
edad        0.097762   0.007963  12.277   <2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8371 on 49 degrees of freedom
Multiple R-squared:  0.7547,    Adjusted R-squared:  0.7497
F-statistic: 150.7 on 1 and 49 DF,  p-value: < 2.2e-16
```

Y listo, obtenemos nuestro modelo de regresión lineal, donde vemos el intercepto, el coeficiente para la edad, la significancia estadística, y los valores de los cuadrados.

1.7.2 Regresión Lineal Múltiple

En este caso, debemos de tener dos o más variables independientes de tipo numérico, asumamos que todas nuestras variables presentan un relación lineal, y creemos el modelo.

```r
set.seed(5)
n <- 50
min_inactividad <- rpois(n, lambda = 60)
calorias <- rpois(n, lambda = 2000)
peso <- rnorm(n, mean = 70, sd = 10)
glucosa <- 80 + 0.3 * min_inactividad - 0.05 * calorias + 1.5 * peso + rnorm(n, mean = 0, sd
```

Estas son las variables, meramente inventadas. La función "rpois" genera una distribución de poisson.

```r
modelo_glucosa <- lm(glucosa ~ min_inactividad + calorias + peso)
summary(modelo_glucosa)
```

```
Call:
lm(formula = glucosa ~ min_inactividad + calorias + peso)

Residuals:
     Min       1Q   Median       3Q      Max
-26.4765  -7.3672   0.1089   7.6880  18.2869

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -56.84471   82.68038  -0.688    0.495
min_inactividad   0.22717    0.20216   1.124    0.267
calorias          0.02456    0.04135   0.594    0.556
```

```
peso              1.38746    0.14600   9.503    2e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.2 on 46 degrees of freedom
Multiple R-squared:  0.6855,    Adjusted R-squared:  0.665
F-statistic: 33.43 on 3 and 46 DF,  p-value: 1.279e-11
```

Y listo, obtenemos los mismos estadísticos pero esta vez por cada variable independiente. Cabe mencionar que primeramente debemos poner nuestra variable dependiente, símbolo de virgulilla, y cada variable independiente que deseemos agregar con un signo de "+".

1.7.3 Regresión Logística Binomial

La regresión logística nos permite crear un modelo entre una variable dependiente binaria y dos o más variables independientes. Hagamos un modelo para intentar predecir si nuestras variables pueden ser útiles para predecir epoc:

```
BD_glm <- BD2
colnames(BD_glm)
```

```
 [1] "id"             "colesterol"     "glucosa"        "hdl"
 [5] "col_hdl_ratio"  "hb1ac"          "diabetes_log"   "lugar_origen"
 [9] "edad"           "sexo"           "altura"         "peso"
[13] "imc"            "act_fisica_fac" "tx_ta"          "tas_1"
[17] "tad_1"          "tas_2"          "tad_2"          "cintura"
[21] "cadera"
```

Vemos las variables de interés, ahora vamos a crear el modelo:

```
modelo <- glm(diabetes_log ~
              col_hdl_ratio +
              edad +
              imc +
              act_fisica_fac +
              cintura +
              cadera,
              data = BD_glm,
              family = binomial)
summary(modelo)
```

```
Call:
glm(formula = diabetes_log ~ col_hdl_ratio + edad + imc + act_fisica_fac +
    cintura + cadera, family = binomial, data = BD_glm)

Coefficients:
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)              -9.160934   1.607063  -5.700 1.20e-08 ***
col_hdl_ratio             0.363060   0.094980   3.822 0.000132 ***
edad                      0.050585   0.010839   4.667 3.06e-06 ***
imc                      -0.018085   0.031703  -0.570 0.568368
act_fisica_facBajo       -0.066731   0.456539  -0.146 0.883790
act_fisica_facSedentario  0.025417   0.503819   0.050 0.959764
cintura                   0.028049   0.021106   1.329 0.183863
cadera                    0.008449   0.020049   0.421 0.673460
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 335.22  on 371  degrees of freedom
Residual deviance: 268.10  on 364  degrees of freedom
  (31 observations deleted due to missingness)
AIC: 284.1

Number of Fisher Scoring iterations: 5
```

La sintaxis es exactamente igual que en la regresión lineal, solamente cambia la función por "glm" y el argumento "family" con la opción de "binomial".

Con eso se crea el modelo, y lo visualizamos con "summary".

Naturalmente, algo que es bastante empleado, es obtener los OR para cada variable predictora, esto asumiendo que presentaron OR significativos.

```
OR_modelo <- exp(coef(modelo))
OR_modelo
```

```
             (Intercept)            col_hdl_ratio                     edad
            0.0001050647             1.4377224851            1.0518859087
                     imc       act_fisica_facBajo act_fisica_facSedentario
            0.9820772213             0.9354469981            1.0257432507
                 cintura                   cadera
            1.0284462851             1.0084845391
```

Naturalmente, puede que solo queramos los significativos, te dejo el siguiente código para hacerlo más fácil:

```
OR_modelo <- exp(coef(modelo))
valores_p <- summary(modelo)$coefficients[, "Pr(>|z|)"]
OR_p_sig <- OR_modelo[valores_p < 0.05 & names(OR_modelo) != "(Intercept)"]
OR_p_sig
```

```
col_hdl_ratio          edad
     1.437722      1.051886
```

1.8 Pruebas Post-Hoc

Se usan posterior a un ANOVA positivo (si se encontraron diferencias) y permiten identificar cuáles grupos específicos presentan la diferencia.

Existen diversas pruebas post hoc, sin embargo, las más empleadas son:

- Tukey
- Bonferroni
- Scheffé

1.8.1 Prueba de Tukey

Es útil cuando se tiene equilibrio entre grupos, se realiza de la siguiente manera:

```
TukeyHSD(ANOVA_test)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = BD2$colesterol ~ BD2$act_fisica_fac)

$`BD2$act_fisica_fac`
                     diff        lwr       upr     p adj
Bajo-Alto       16.042898   3.465897 28.619898 0.0080425
Sedentario-Alto 10.504854  -3.736787 24.746496 0.1933351
Sedentario-Bajo -5.538043 -18.115044  7.038957 0.5545997
```

La ventaja de dicha prueba, es que nos otorga la diferencia entre grupos (diff), los intervalos de confianza inferior (lwr) y superior (upr), y el valor de p. Con lo anterior constatamos (aún más…), de que no hay diferencia entre las edad de los grupos. Si en la prueba de ANOVA hubiera resultado positiva, aquí sabríamos cuál.

1.8.2 Prueba de Bonferroni

Útil para pequeños grupos, ya que se ajusta al número de comparaciones, pero es más estricto, lo que genera resultados significativos más certeros. Se hace de la siguiente manera:

```
pairwise.t.test(BD2$edad, BD2$act_fisica_fac, p.adjust.method = "bonferroni")
```

```
    Pairwise comparisons using t tests with pooled SD

data:  BD2$edad and BD2$act_fisica_fac

           Alto    Bajo
Bajo       0.20206 -
Sedentario 3.7e-06 0.00065

P value adjustment method: bonferroni
```

Y nos arroja los resultados en una tabla cruzada con los valores de p.

1.8.3 Prueba de Scheffé

Es útil cuando se observan grupos desiguales. Se hace de la siguiente forma:

```
pacman::p_load(DescTools)
ScheffeTest(ANOVA_test)
```

```
  Posthoc multiple comparisons of means: Scheffe Test
    95% family-wise confidence level

$`BD2$act_fisica_fac`
                     diff     lwr.ci    upr.ci   pval
Bajo-Alto       16.042898   2.907358 29.178437 0.0117 *
Sedentario-Alto 10.504854  -4.369252 25.378960 0.2231
Sedentario-Bajo -5.538043 -18.673583  7.597496 0.5851
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Y al igual que con Tukey, nos otorga la diferencia, IC y valores de p.