

TIPOLOGIA Y CICLO DE VIDA DE LOS DATOS

PRACTICA 2.

Victor Manuel Vásquez Rivas - Francisco Javier Moreno Hernández

Mayo 2019

- 1 Descripción, limpieza y tratamiento en los datos
 - 1.1 Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?
 - 1.2 Integración y selección de los datos de interés a analizar.
 - 1.3 Limpieza de los datos.
- 2 Análisis y pruebas estadísticas
 - 2.1 Análisis de los datos.
 - 2.2 Representación de los resultados a partir de tablas y gráficas.
 - 2.3 Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?
 - 2.4 Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.
 - 2.5 Referencias

1 Descripción, limpieza y tratamiento en los datos

1.1 Descripción del dataset. ¿Por qué es importante y qué pregunta/problema pretende responder?

El data set que hemos seleccionado, llamado “heart-disease-uci”, o lo que es lo mismo, “enfermedades-corazon-uci”, se refiere a pacientes. Tenemos una lista de atributos con características de los pacientes, como son edad, sexo, colesterol en sangre, máximo ritmo cardíaco y otros indicadores médicos. La variable “target” indica si el paciente tiene enfermedad cardíaca o no y es la variable sobre la que querremos hacer predicciones.

Hemos encontrado el data set interesante porque aunque los datos en el csv son numéricos ya vemos de un principio que hay datos categóricos.

Por otra parte, la variable “target” parece que nos va a permitir realizar estudios de diferente tipo como son: clasificación de pacientes, regresiones para predecir el valor, etc.

```
diseases<- read.csv("input/heart.csv", sep=",", na.strings = "NA")  
dim(diseases)
```

```
## [1] 303 14
```

```
str(diseases)
```

```
## 'data.frame': 303 obs. of 14 variables:
## $ age : int 63 37 41 56 57 57 56 44 52 57 ...
## $ sex : int 1 1 0 1 0 1 0 1 1 1 ...
## $ cp : int 3 2 1 1 0 0 1 1 2 2 ...
## $ trestbps: int 145 130 130 120 120 140 140 120 172 150 ...
## $ chol : int 233 250 204 236 354 192 294 263 199 168 ...
## $ fbs : int 1 0 0 0 0 0 0 0 1 0 ...
## $ restecg : int 0 1 0 1 1 1 0 1 1 1 ...
## $ thalach : int 150 187 172 178 163 148 153 173 162 174 ...
## $ exang : int 0 0 0 0 1 0 0 0 0 0 ...
## $ oldpeak : num 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
## $ slope : int 0 0 2 2 2 1 1 2 2 2 ...
## $ ca : int 0 0 0 0 0 0 0 0 0 0 ...
## $ thal : int 1 2 2 2 2 1 2 3 3 2 ...
## $ target : int 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(diseases)
```

```
##      age      sex      cp      trestbps
## Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.    : 94.0
## 1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0
## Median :55.00   Median :1.0000   Median :1.000   Median :130.0
## Mean   :54.37   Mean    :0.6832   Mean    :0.967   Mean    :131.6
## 3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0
## Max.    :77.00   Max.     :1.0000   Max.     :3.000   Max.     :200.0
##      chol      fbs      restecg      thalach
## Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.    : 71.0
## 1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5
## Median :240.0   Median :0.0000   Median :1.0000   Median :153.0
## Mean   :246.3   Mean    :0.1485   Mean    :0.5281   Mean    :149.6
## 3rd Qu.:274.5   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:166.0
## Max.    :564.0   Max.     :1.0000   Max.     :2.0000   Max.     :202.0
##      exang      oldpeak      slope      ca
## Min.   :0.0000   Min.   :0.00   Min.   :0.000   Min.    :0.0000
## 1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
## Median :0.0000   Median :0.80   Median :1.000   Median :0.0000
## Mean   :0.3267   Mean    :1.04   Mean    :1.399   Mean    :0.7294
## 3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
## Max.    :1.0000   Max.     :6.20   Max.     :2.000   Max.     :4.0000
##      thal      target
## Min.   :0.000   Min.   :0.0000
## 1st Qu.:2.000   1st Qu.:0.0000
## Median :2.000   Median :1.0000
## Mean   :2.314   Mean    :0.5446
## 3rd Qu.:3.000   3rd Qu.:1.0000
## Max.    :3.000   Max.     :1.0000
```

1.2 Integración y selección de los datos de interés a analizar.

A continuación se describirán el significado de cada una de las variables del data set:

- age: edad en años.
- sex: 1 = masculino; 0 = femenino.
- cp: tipo de dolor de pecho.
- trestbps: presión arterial en reposo (en mm Hg al ingreso en el hospital).
- chol: suero colestoral en mg/dl.
- fbs: azúcar en la sangre en ayunas > 120 mg/dl (1 = verdadero; 0 = falso).
- restecg: resultados de electrocardiograma en reposo.
- thalach: maximo ritmo cardiaco alcanzado
- exang: angina inducida por el ejercicio (1 = sí; 0 = no)
- oldpeak: depresión del ST inducida por el ejercicio en relación con el descanso.
- slope: Pendiente maxima del segmento del ejercicio ST.
- ca: número de vasos principales (0-3) coloreados por fluoroscopia.
- thal: 3 = normal; 6 = defecto fijo; 7 = defecto reversible
- target: 1 o 0

Para la selección de los atributos, comenzaremos revisando las relaciones entre los distintos atributos y el atributo target el cual indica si existe o no un problema cardiaco, así diagnosticar si son representativos o pueden ser eliminados.

```
#Cargamos antes algunas librerías que utilizaremos  
if(!require(ggplot2)){  
  install.packages('ggplot2', repos='http://cran.us.r-project.org')  
  library(ggplot2)  
}
```

```
## Loading required package: ggplot2
```

```
if(!require(grid)){  
  install.packages('grid', repos='http://cran.us.r-project.org')  
  library(grid)  
}
```

```
## Loading required package: grid
```

```
if(!require(gridExtra)){  
  install.packages('gridExtra', repos='http://cran.us.r-project.org')  
  library(gridExtra)  
}
```

```
## Loading required package: gridExtra
```

```
if(!require(fBasics)){  
  install.packages('fBasics', repos='http://cran.us.r-project.org')  
  library(fBasics)  
}
```

```
## Loading required package: fBasics
```

```
## Loading required package: timeDate
```

```
## Loading required package: timeSeries
```

#Trasformaremos algunos datos para poder graficar las variables y encontrar las de interés

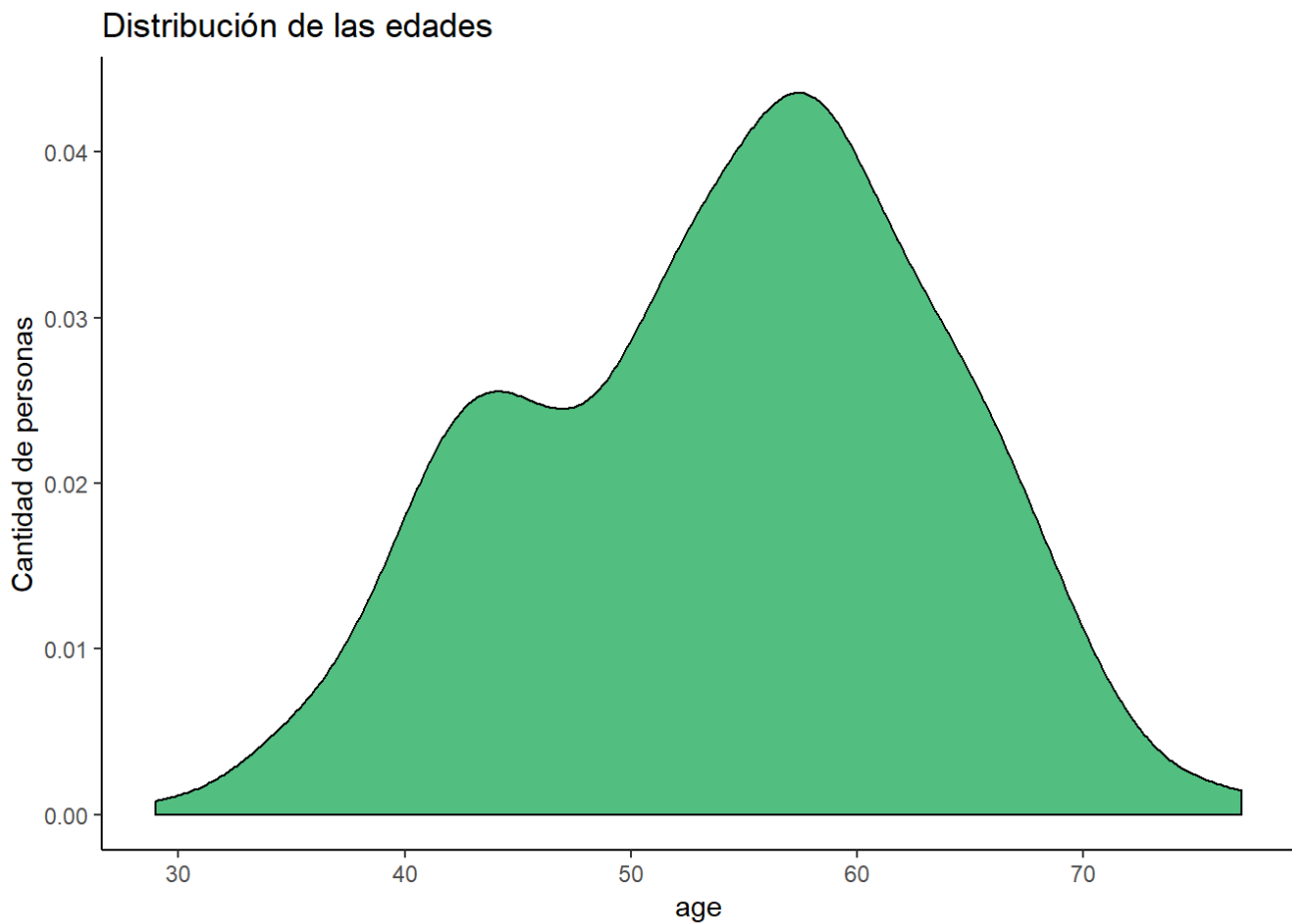
```
diseases$target.tipo[diseases$target == 0 ]="Sin Problemas Cardiacos"
diseases$target.tipo[diseases$target > 0 ]="Con Problemas Cardiacos"
diseases$age.tipo[diseases$age < 40 ]="Menores a 40"
diseases$age.tipo[diseases$age >= 40 & diseases$age < 50 ]="Mayor a 40 y menor a 50"
diseases$age.tipo[diseases$age >= 50 & diseases$age < 60 ]="Mayor a 50 y menor a 60"
diseases$age.tipo[diseases$age >= 60 & diseases$age < 70 ]="Mayor a 60 y menor a 70"
diseases$age.tipo[diseases$age >= 70 ]="Mayores a 70"
diseases$sex.tipo[diseases$sex == 0 ] = "Femenino"
diseases$sex.tipo[diseases$sex == 1 ] = "Masculino"
diseases$cp.tipo = as.factor(diseases$cp)
diseases$trestbps.tipo[diseases$trestbps < 100] = "Menor a 100"
diseases$trestbps.tipo[diseases$trestbps >= 100 & diseases$trestbps < 140] = "Mayor a 100 y menor a 150"
diseases$trestbps.tipo[diseases$trestbps >= 140 & diseases$trestbps < 180] = "Mayor a 140 y menor a 180"
diseases$trestbps.tipo[diseases$trestbps >= 180] = "Mayor a 180"
diseases$chol.tipo[diseases$chol < 200] = "Menor a 200"
diseases$chol.tipo[diseases$chol >= 200 & diseases$chol < 300] = "Mayor a 200 y menor a 300"
diseases$chol.tipo[diseases$chol >= 300 & diseases$chol < 400] = "Mayor a 300 y menor a 400"
diseases$chol.tipo[diseases$chol >= 400] = "Mayor a 400"
diseases$fbs.tipo = as.factor(diseases$fbs)
diseases$restecg.tipo = as.factor(diseases$restecg)
diseases$thalach.tipo[diseases$thalach < 100] = "Menor a 100"
diseases$thalach.tipo[diseases$thalach >= 100 & diseases$thalach < 150] = "Mayor a 100 y menor a 150"
diseases$thalach.tipo[diseases$thalach >= 150 & diseases$thalach < 200] = "Mayor a 150 y menor a 200"
diseases$thalach.tipo[diseases$thalach >= 200] = "Mayor a 200"
diseases$exang.tipo = as.factor(diseases$exang)
diseases$oldpeak.tipo[diseases$oldpeak < 1] = "Menor a 1"
diseases$oldpeak.tipo[diseases$oldpeak >= 1 & diseases$oldpeak < 2] = "Mayor a 1 y menor a 2"
diseases$oldpeak.tipo[diseases$oldpeak >= 2 & diseases$oldpeak < 3] = "Mayor a 2 y menor a 3"
diseases$oldpeak.tipo[diseases$oldpeak >= 3 & diseases$oldpeak < 4] = "Mayor a 3 y menor a 4"
diseases$oldpeak.tipo[diseases$oldpeak >= 4 & diseases$oldpeak < 5] = "Mayor a 4 y menor a 5"
diseases$oldpeak.tipo[diseases$oldpeak >= 5] = "Mayor a 5"
diseases$slope.tipo = as.factor(diseases$slope)
diseases$ca.tipo = as.factor(diseases$ca)
```

```
diseases$thal.tipo = as.factor(diseases$thal)
```

```
# Distribución de las edades
```

```
ggplot(diseases,aes(x = age)) + geom_density(bins =30,fill ="#52BE80") + theme_bw()  
+ theme_classic() +ggtitle("Distribución de las edades") +ylab("Cantidad de person  
as")
```

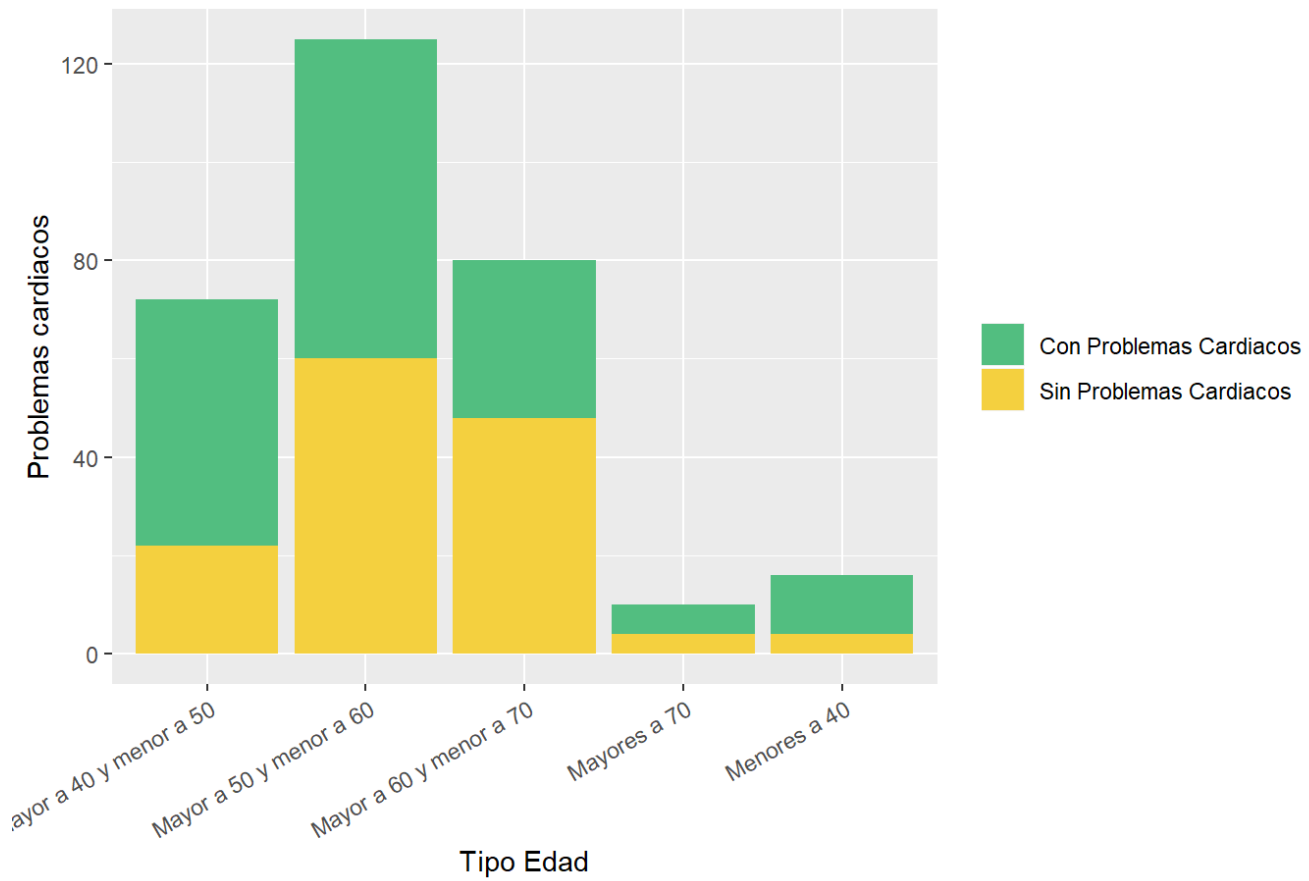
```
## Warning: Ignoring unknown parameters: bins
```



```
# Distribución de problemas cardiacos por edad
```

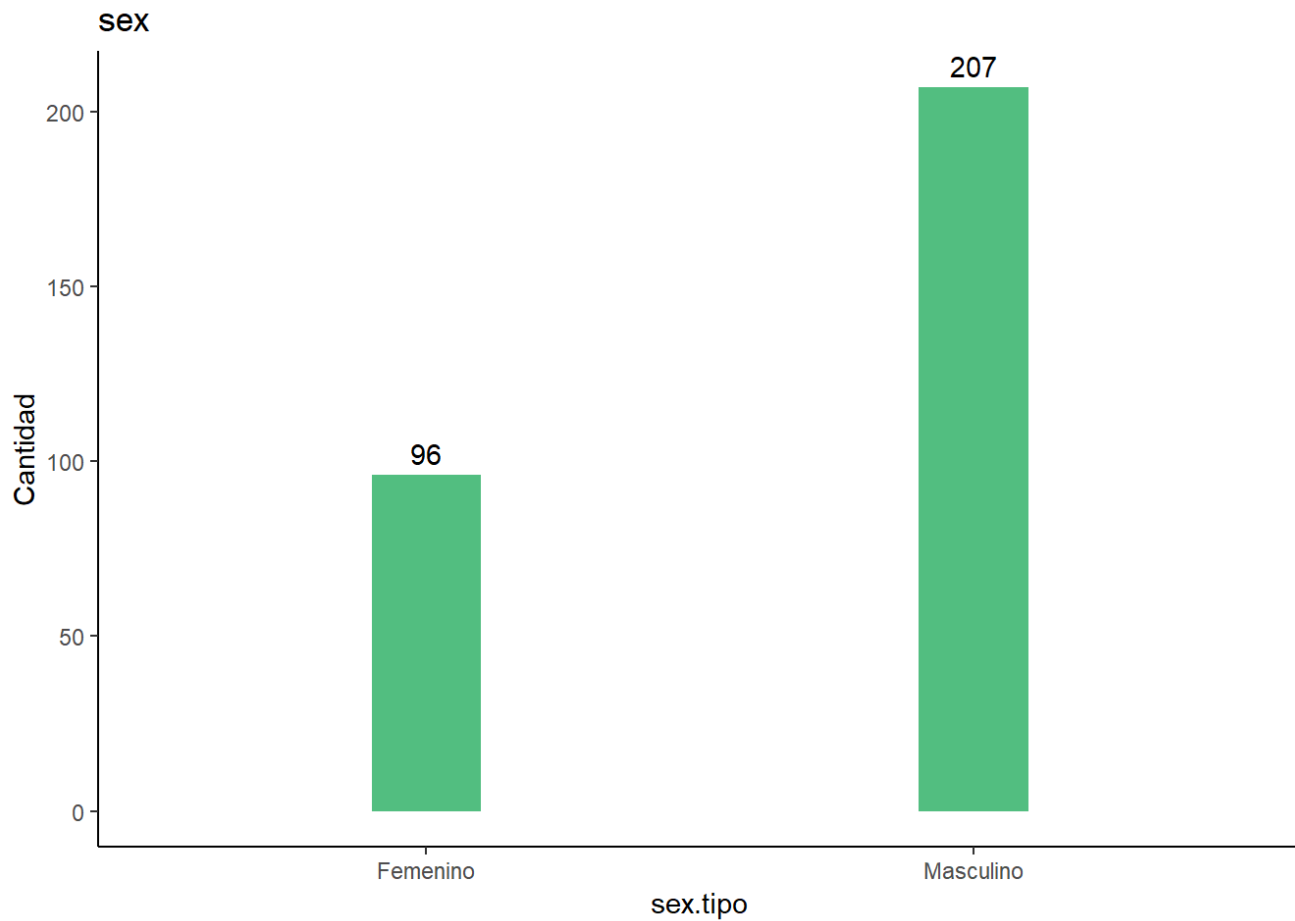
```
ggplot(diseases,aes(age.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo Edad", y=  
"Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas car  
dacos por edades")+ scale_fill_manual(values=c("#52BE80","#F4D03F")) + theme(axis.  
text.x = element_text(angle = 30, hjust = 1))
```

Problemas cardiacos por edades



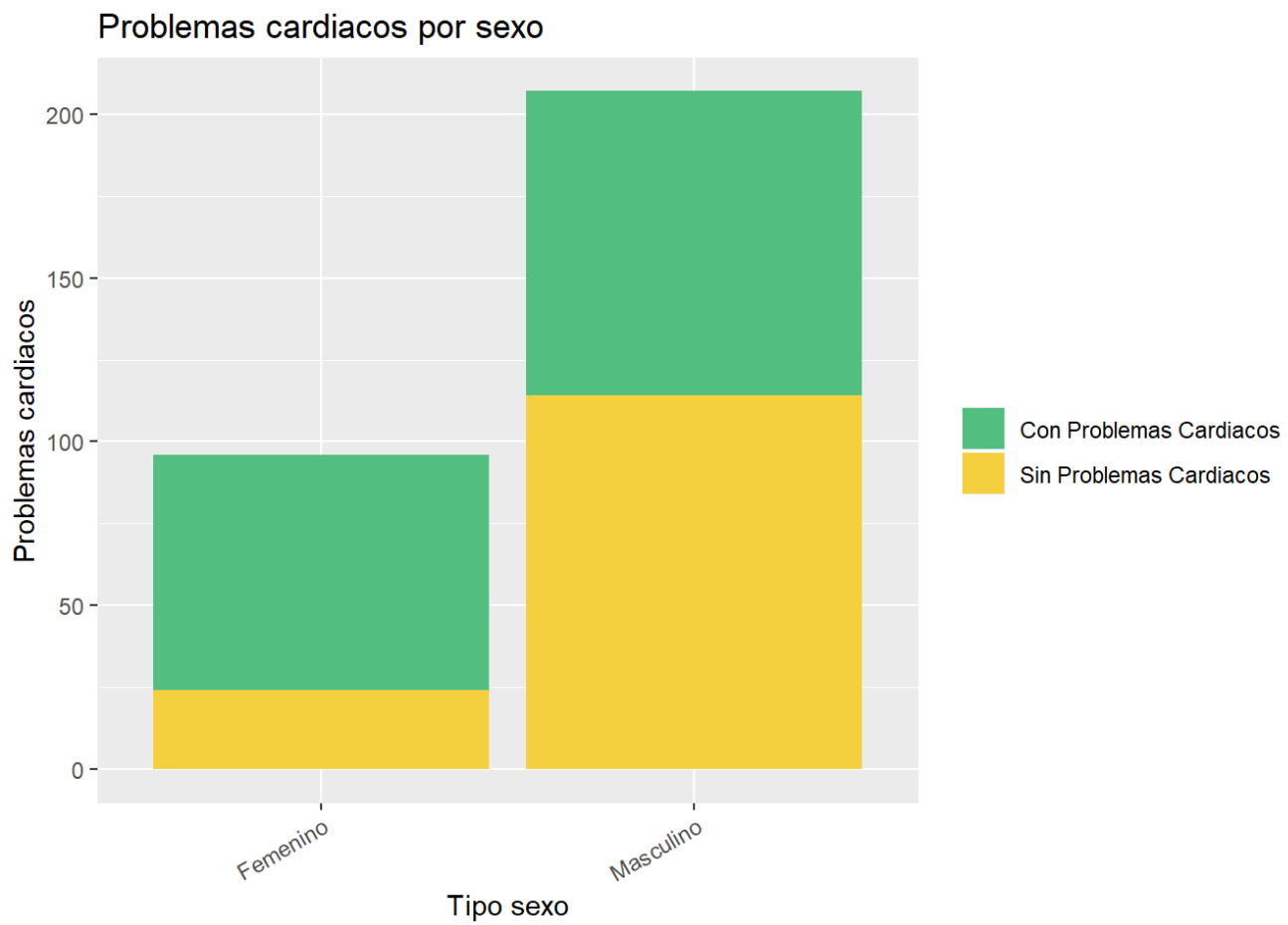
Distribución por sexo

```
ggplot(diseases,aes(x =sex.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic() +ylab("Cantidad") + ggtitle("sex")
```

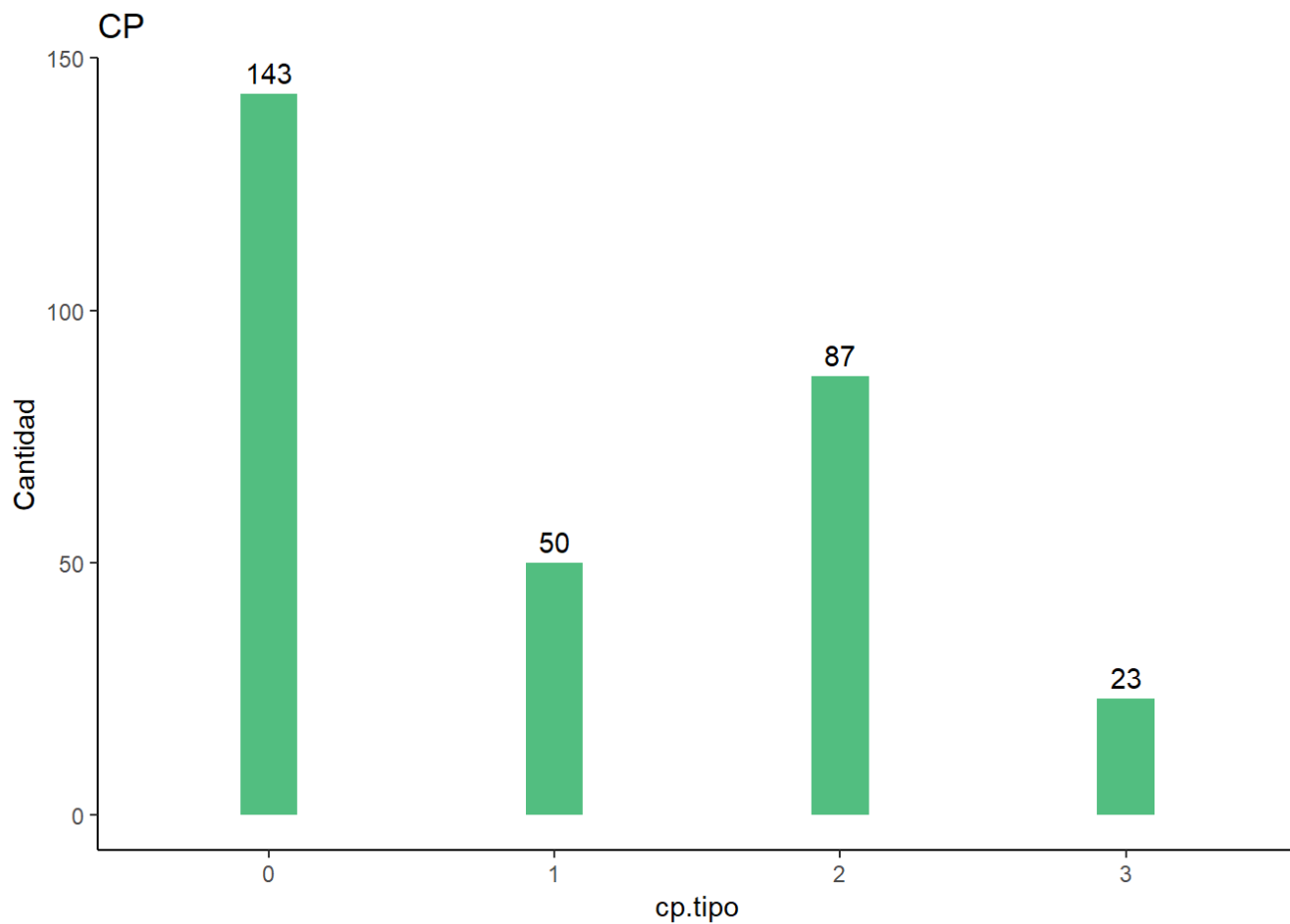


Distribución de problemas cardiacos por sexo

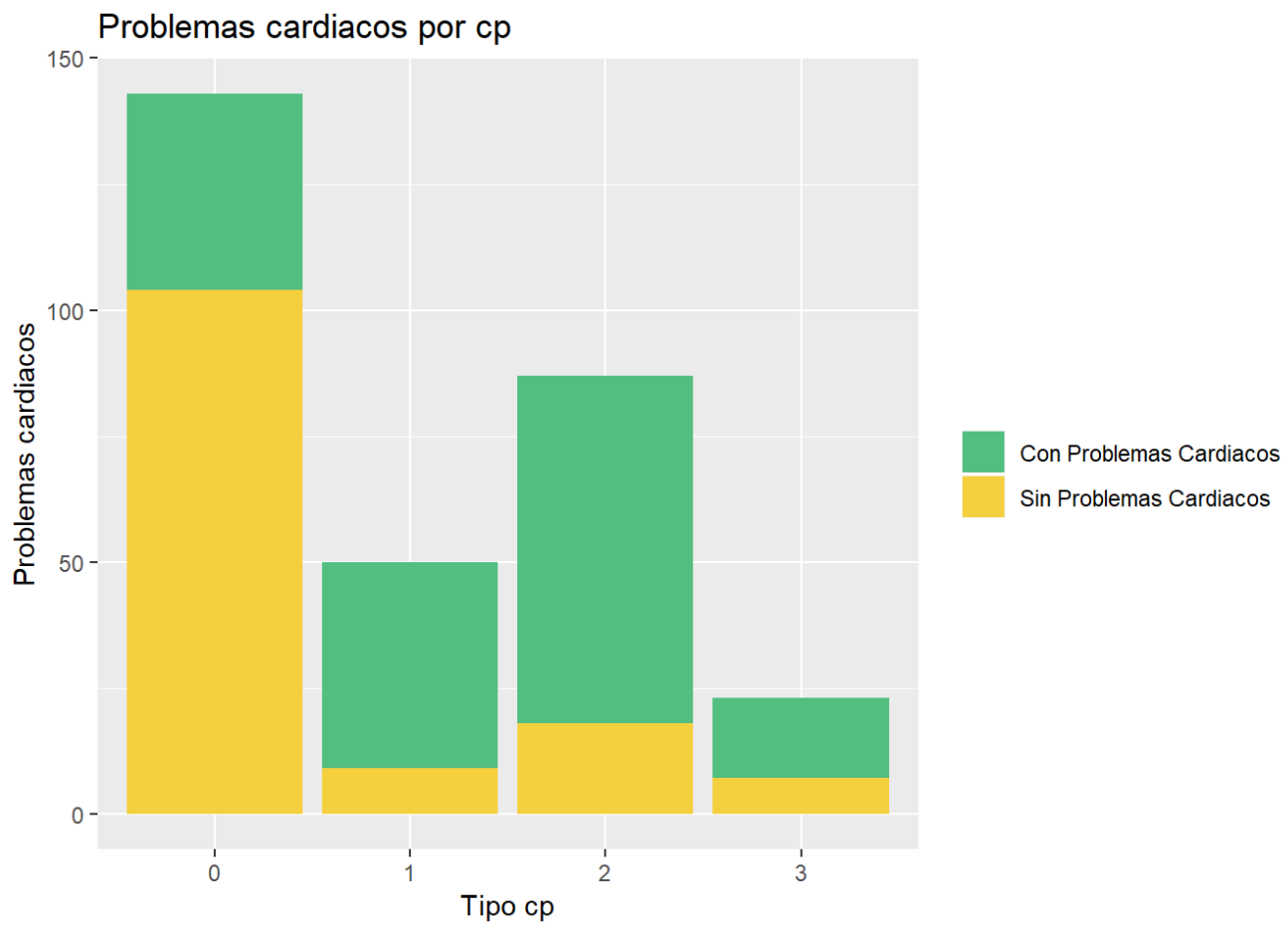
```
ggplot(diseases,aes(sex.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo sexo", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardiacos por sexo")+ scale_fill_manual(values=c("#52BE80","#F4D03F")) + theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

```
# Distribución por CP
ggplot(diseases,aes(x =cp.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(
  stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic()
+ylab("Cantidad") + ggtitle("CP")
```

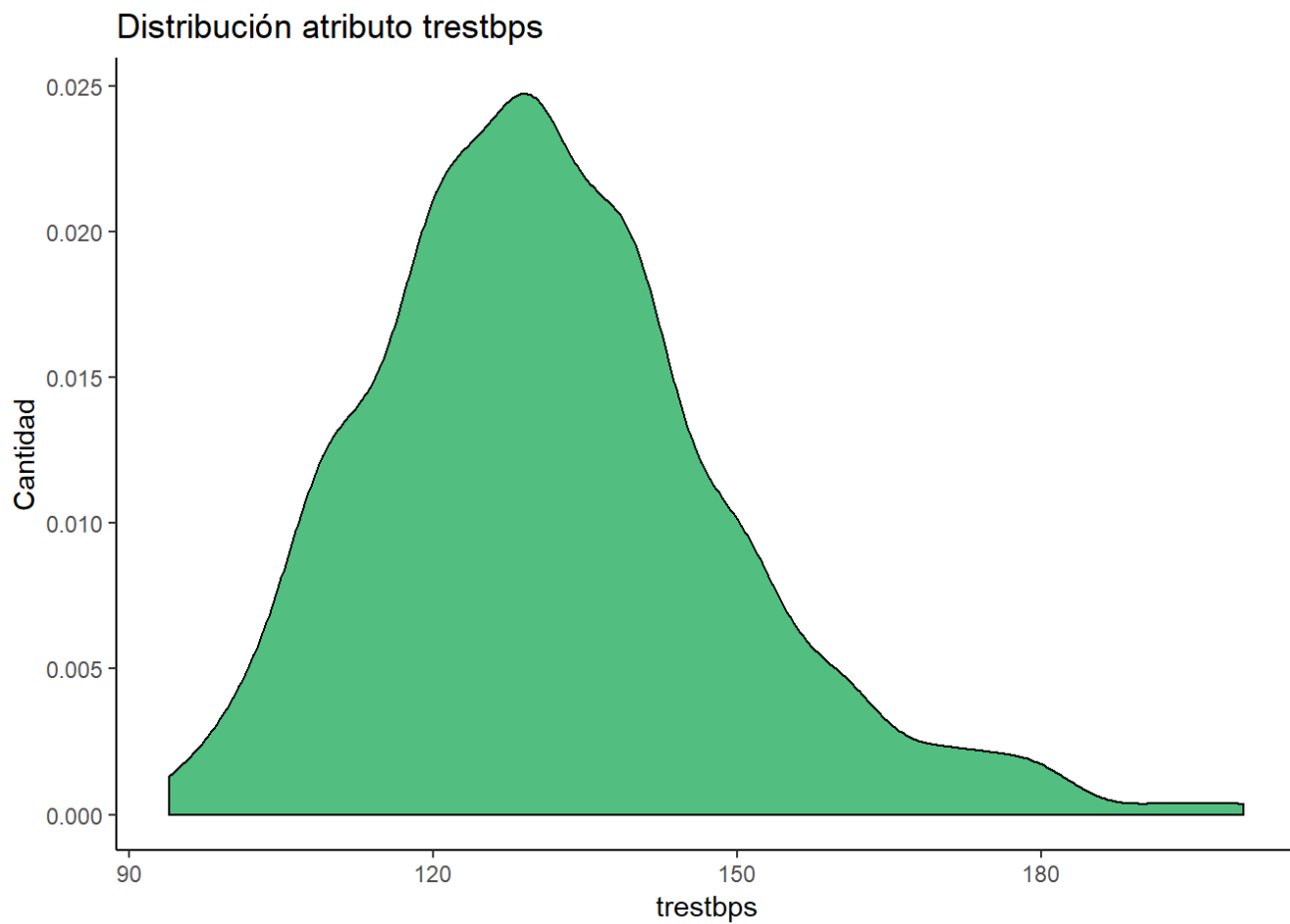


```
# Distribución de problemas cardiacos por CP
#Esta es una variable importante en donde podemos ver que si el tipo de CP es 0 existen menos probabilidades que tenga un problema cardiaco.
ggplot(diseases,aes(cp.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo cp", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardiacos por cp")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```

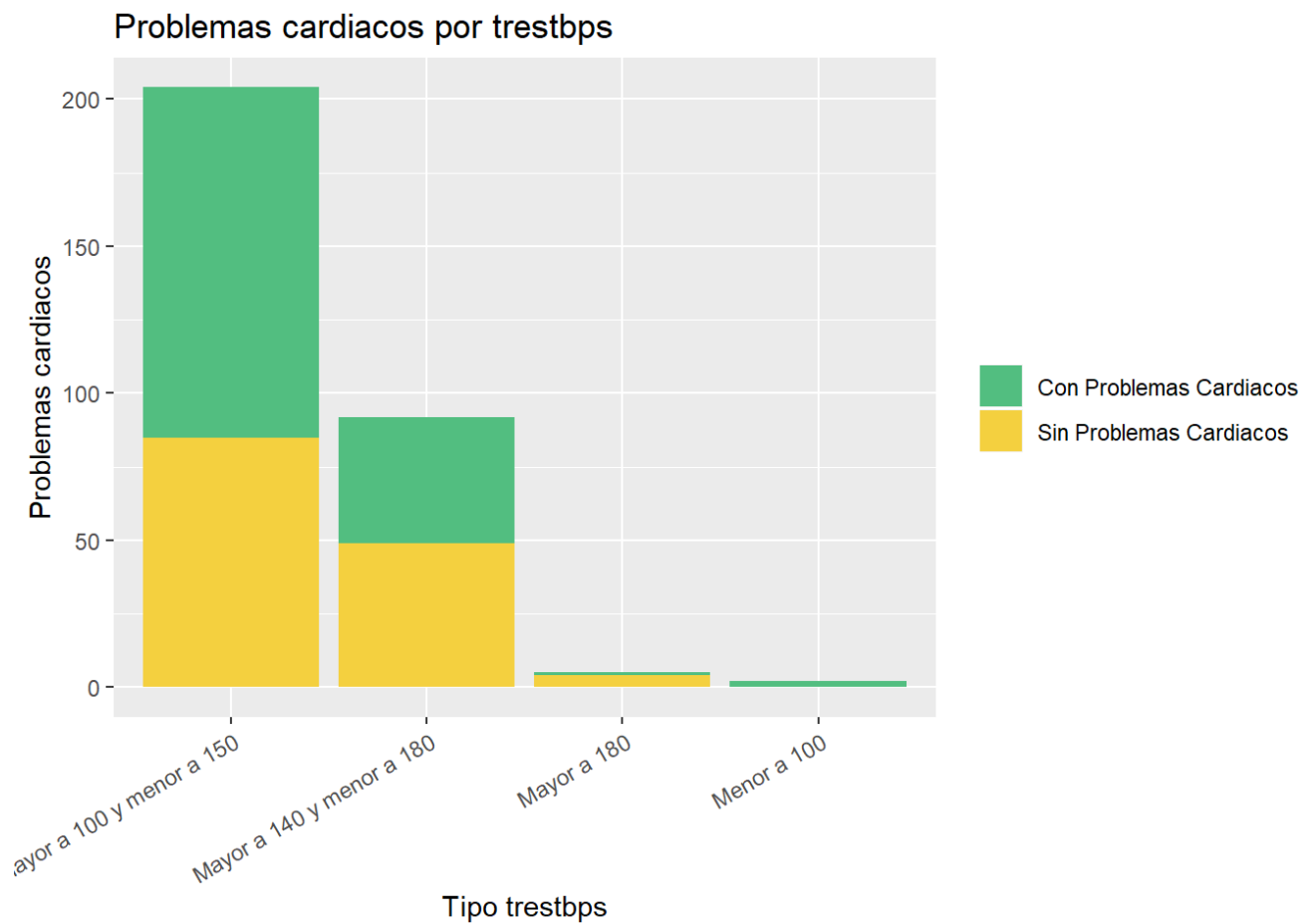


```
# Distribución por trestbps
ggplot(diseases,aes(x = trestbps)) + geom_density(bins =30,fill ="#52BE80") + theme
_bw() + theme_classic() +ggtitle("Distribución atributo trestbps") + ylab("Cantida
d")
```

```
## Warning: Ignoring unknown parameters: bins
```



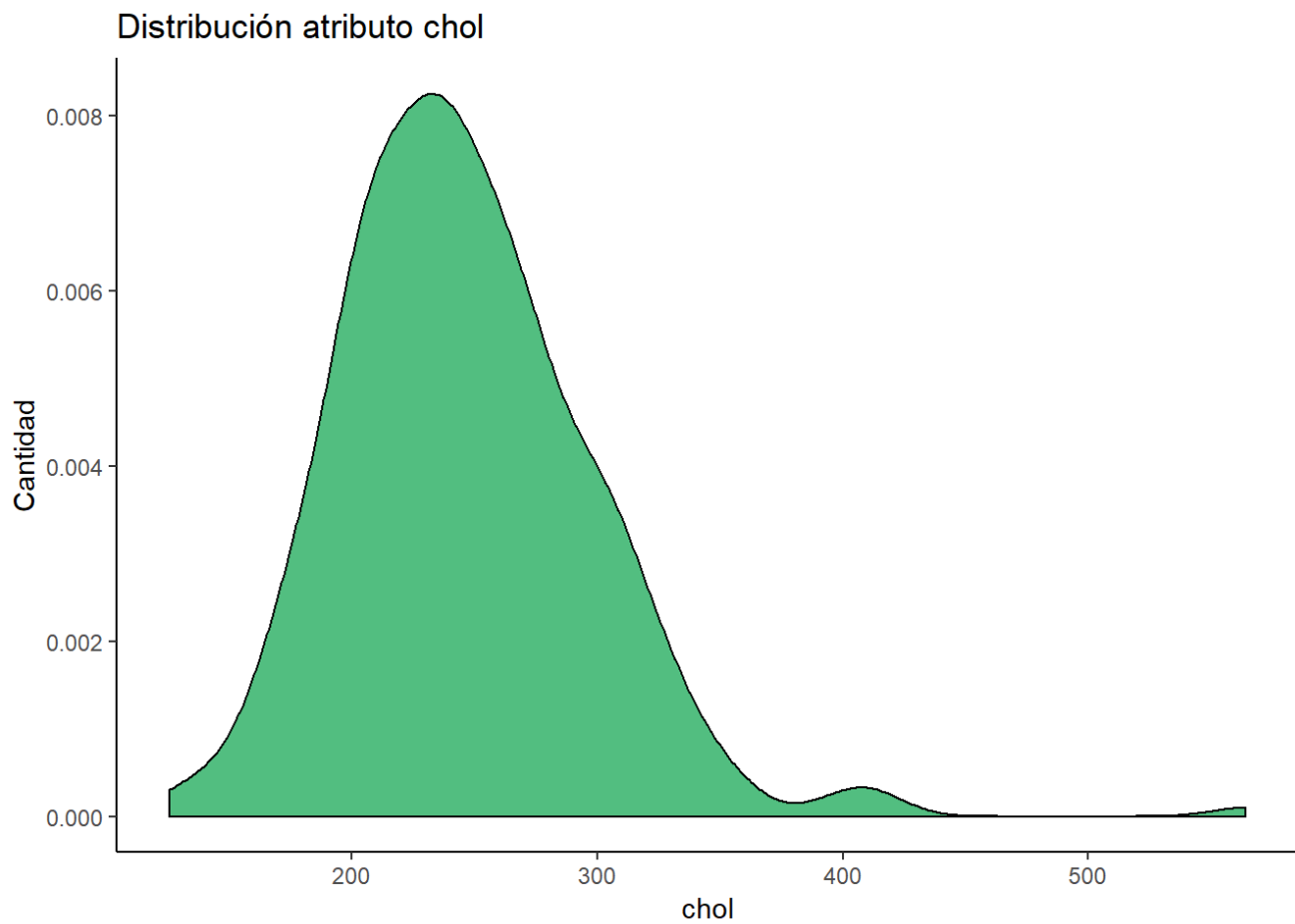
```
# Distribución de problemas cardiacos por trestbps
ggplot(diseases,aes(trestbps.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo trest
bps", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Probl
emas cardiacos por trestbps")+ scale_fill_manual(values=c("#52BE80","#F4D03F")) + t
heme(axis.text.x = element_text(angle = 30, hjust = 1))
```



```
# Distribución chol
```

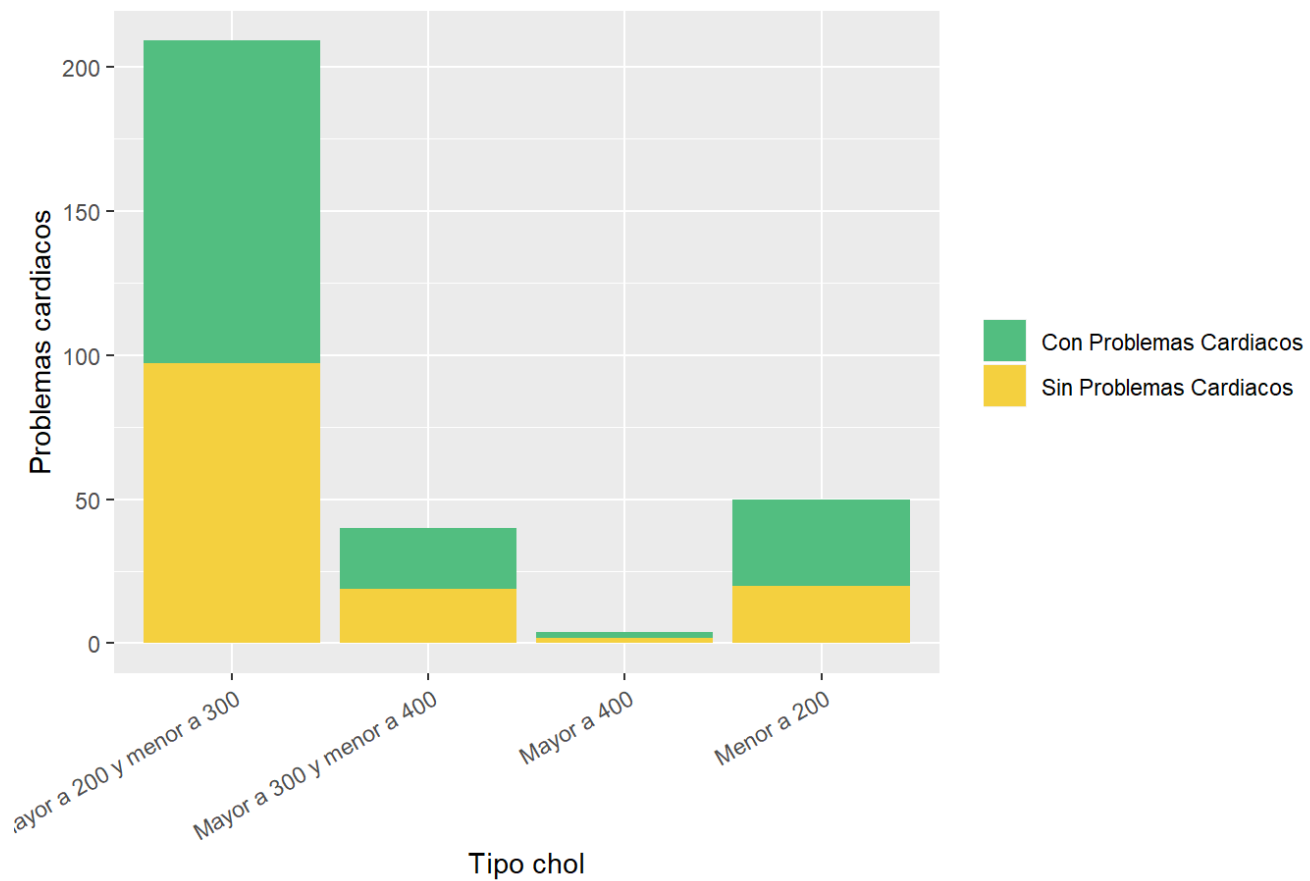
```
ggplot(diseases,aes(x = chol)) + geom_density(bins =30,fill ="#52BE80") + theme_bw  
( ) + theme_classic() +ggtitle("Distribución atributo chol") + ylab("Cantidad")
```

```
## Warning: Ignoring unknown parameters: bins
```



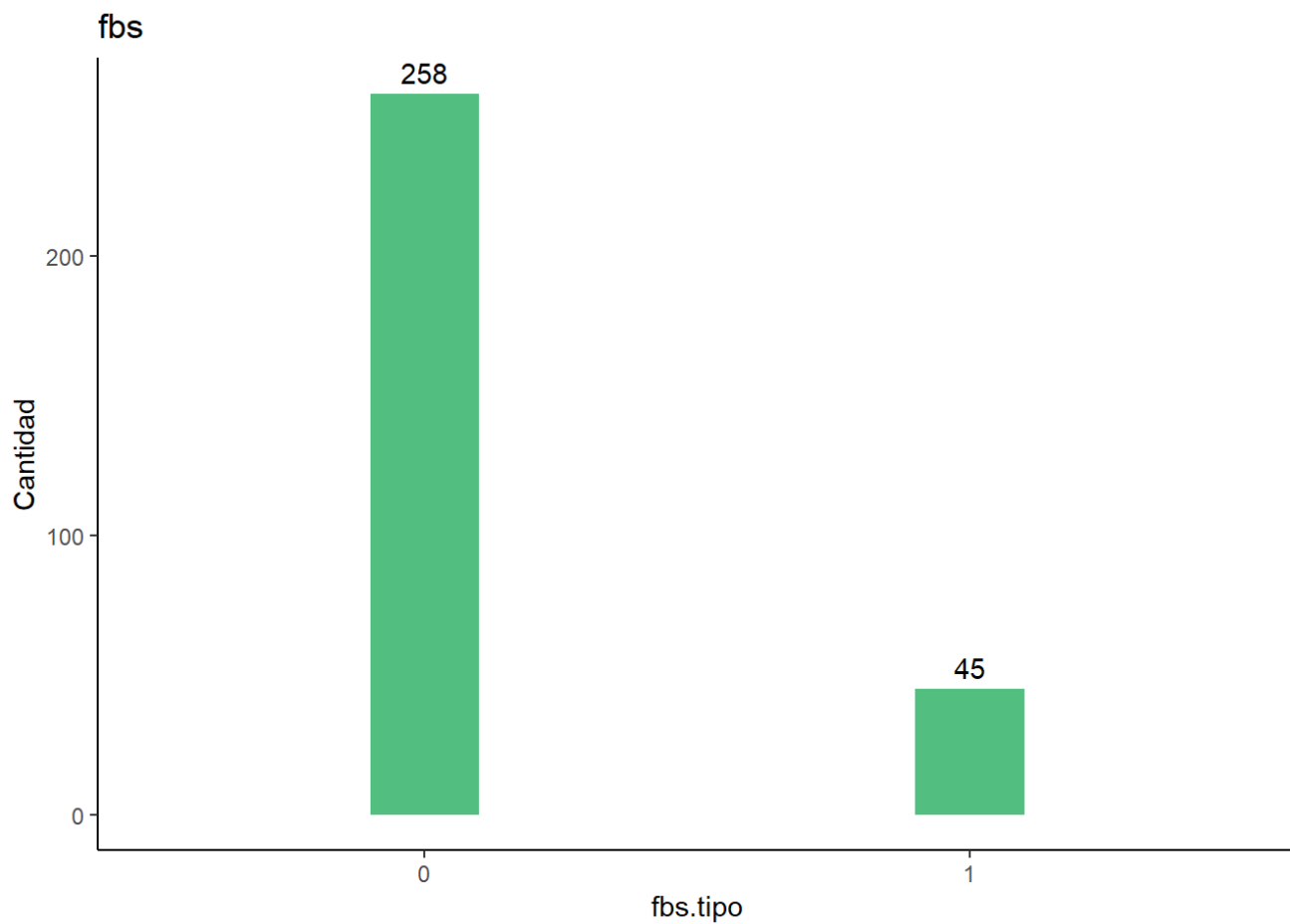
```
# Distribución de problemas cardiacos por Chol
ggplot(diseases,aes(chol.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo chol", y=
"Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas car
diacos por chol")+ scale_fill_manual(values=c("#52BE80","#F4D03F")) + theme(axis.te
xt.x = element_text(angle = 30, hjust = 1))
```

Problemas cardiacos por chol



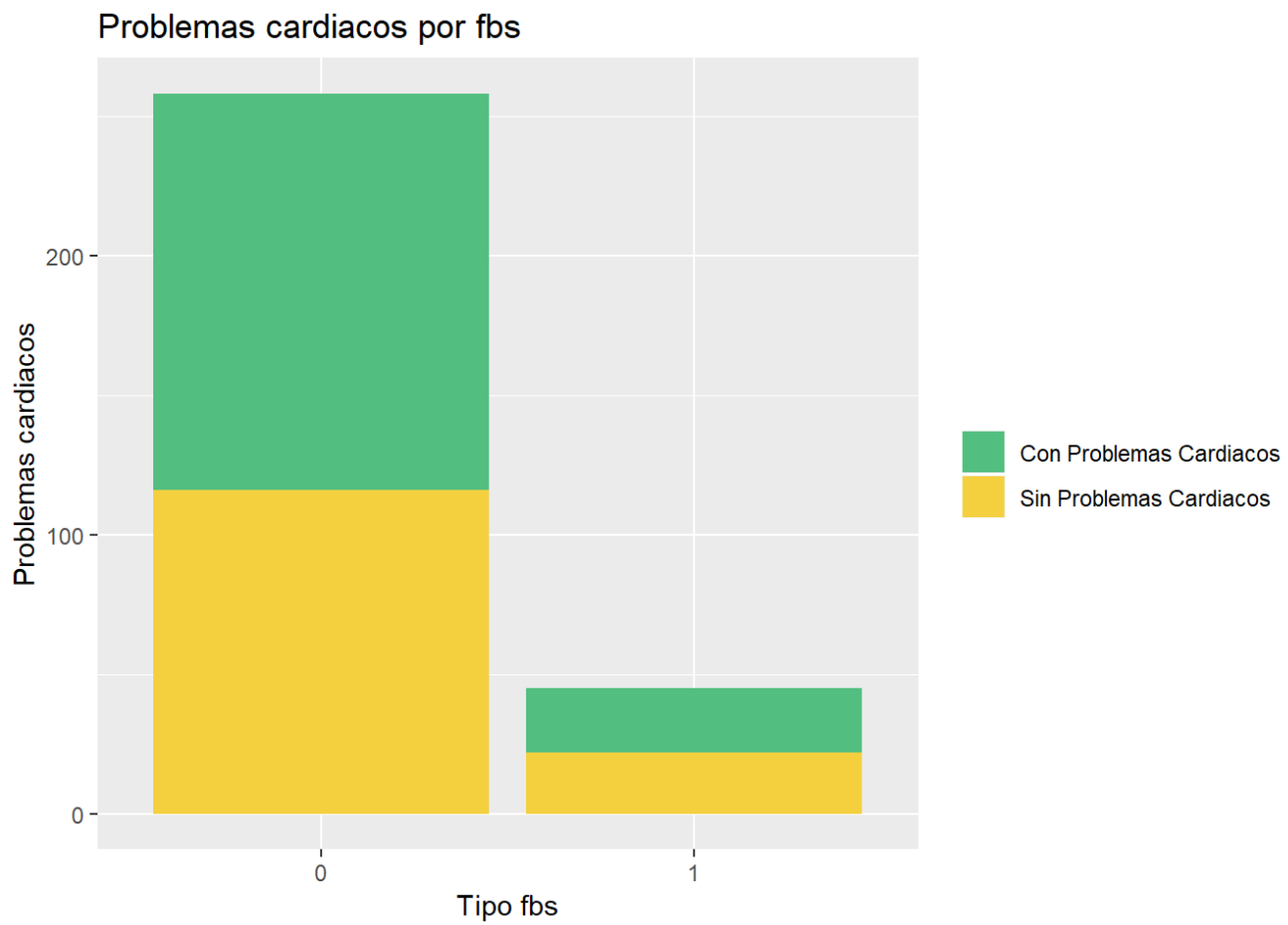
Distribución fbs

```
ggplot(diseases,aes(x =fbs.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic() +ylab("Cantidad") + ggtitle("fbs")
```



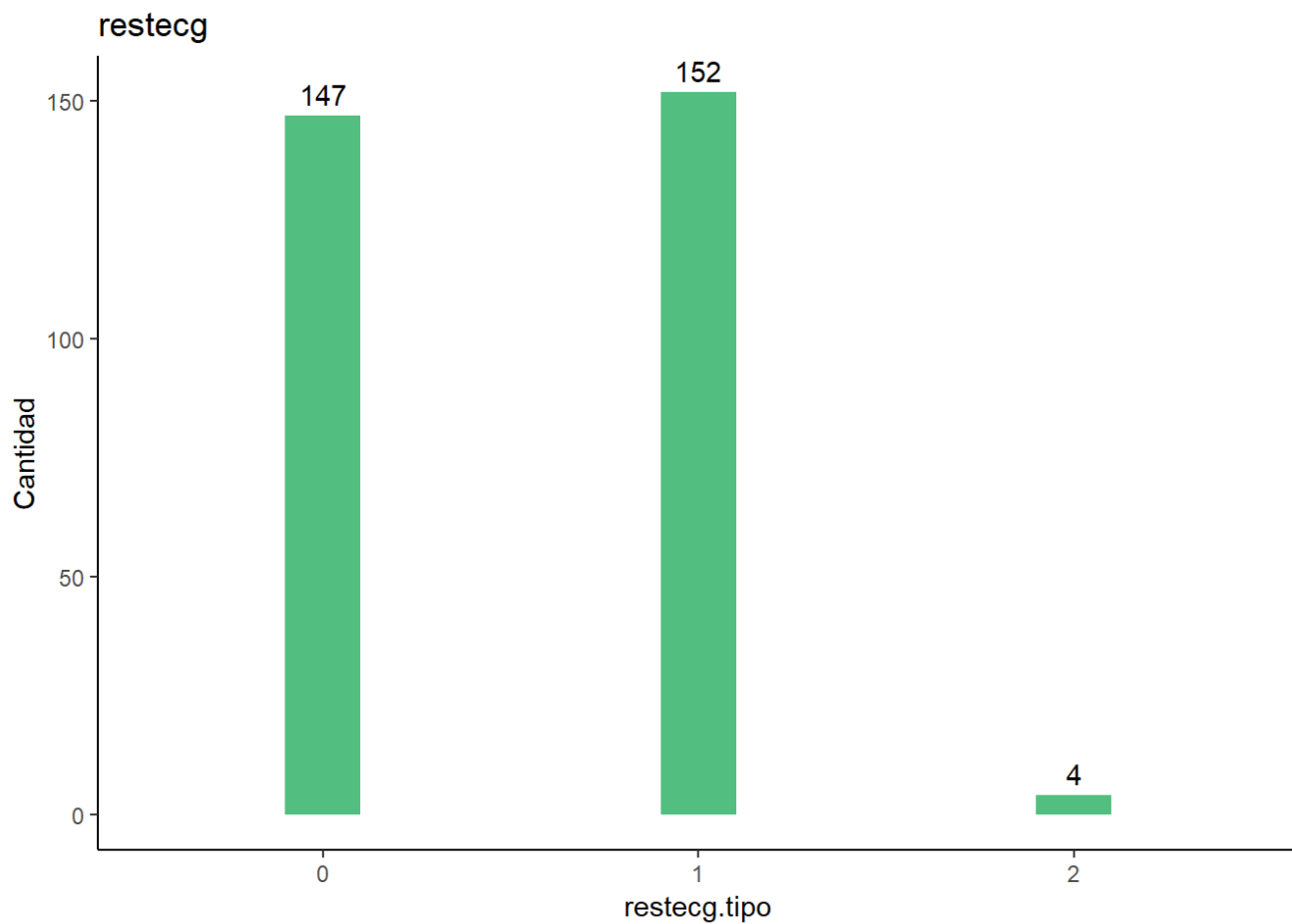
```
# Distribución de problemas cardiacos por fbs
```

```
ggplot(diseases,aes(fbs.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo fbs", y="P  
roblemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardi  
acos por fbs")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```

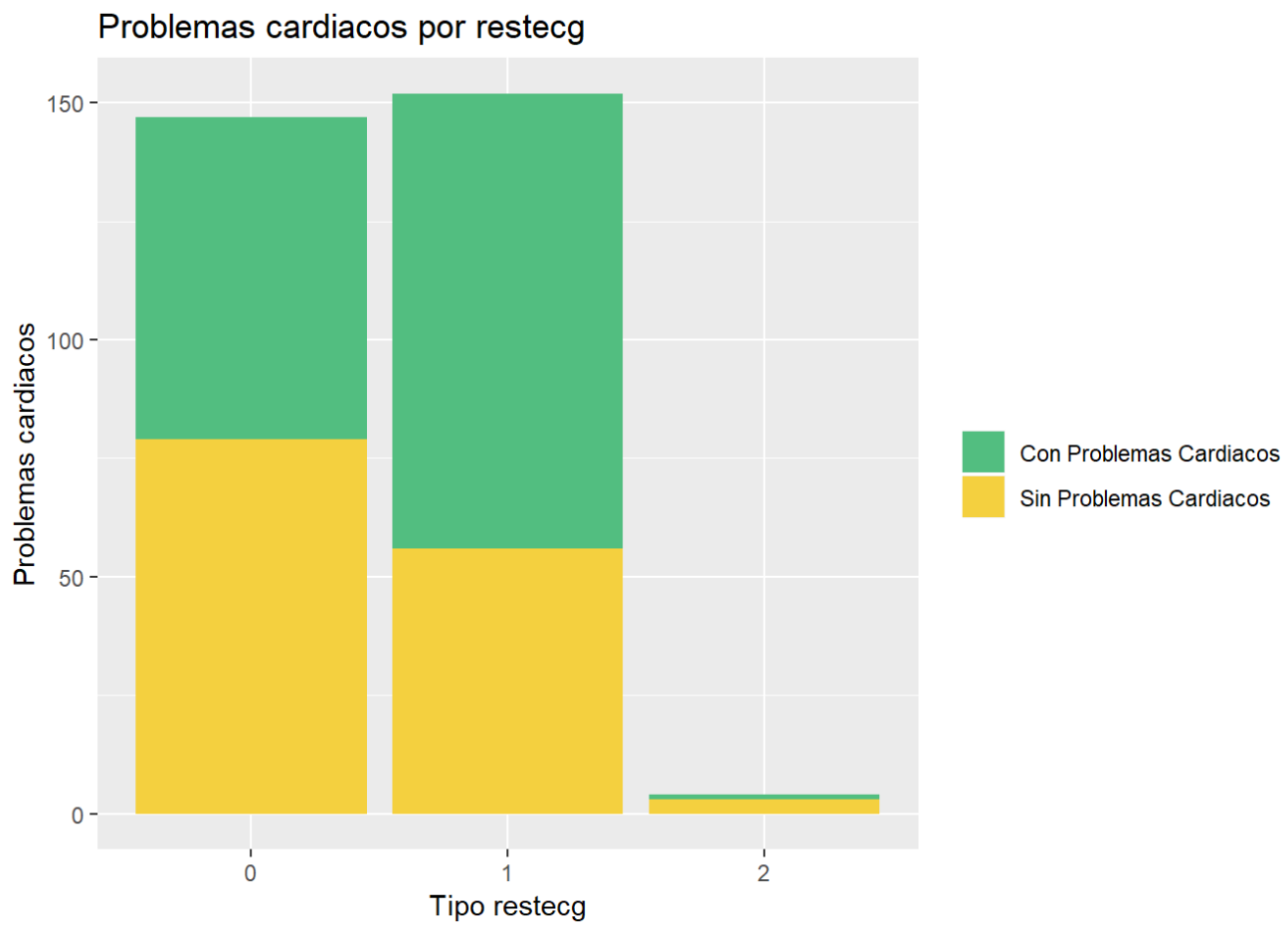



```
# Distribución restecg
```

```
ggplot(diseases,aes(x =restecg.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic() +ylab("Cantidad") + ggtitle("restecg")
```



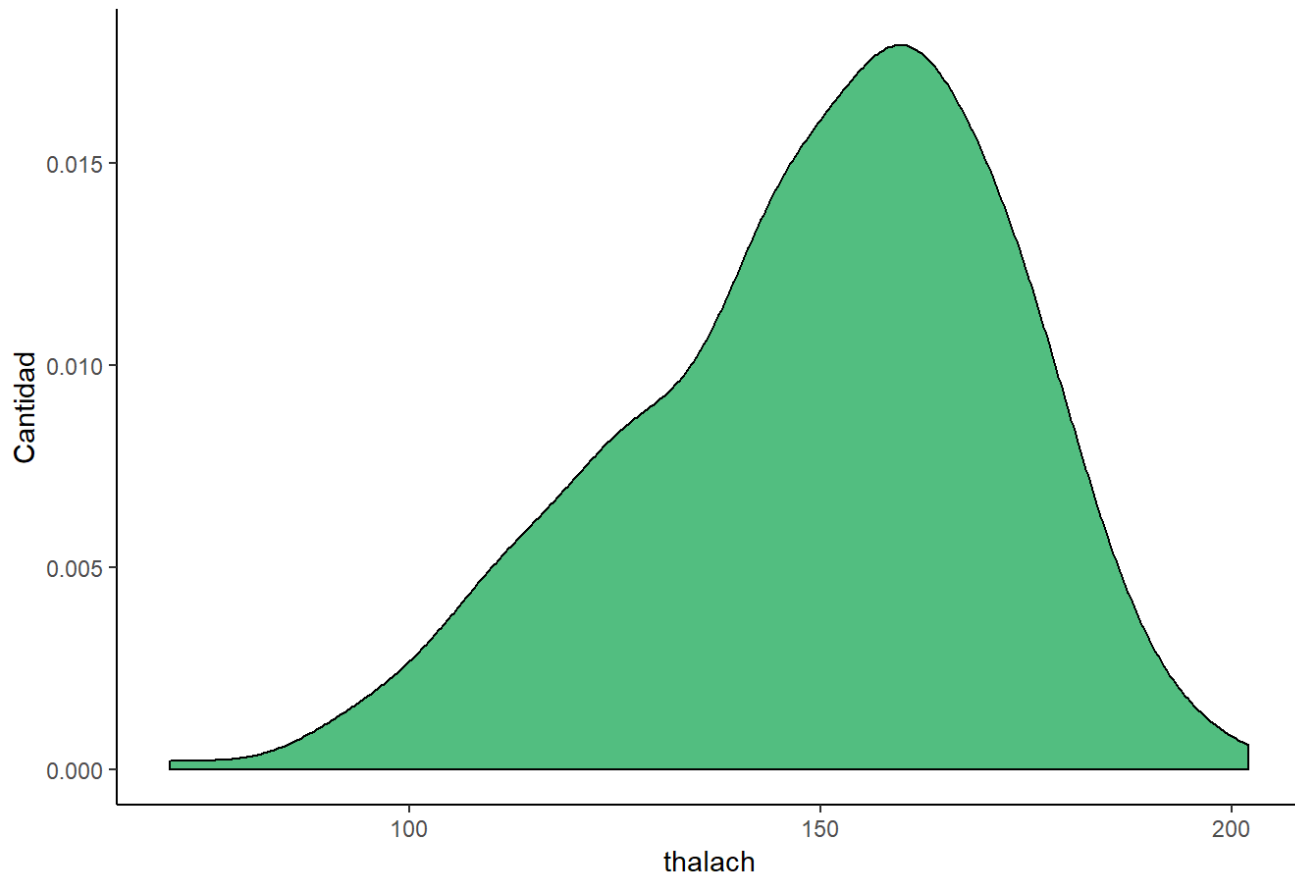
```
# Distribución de problemas cardiacos por restecg
ggplot(diseases,aes(restecg.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo restecg", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardiacos por restecg")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```



```
# Distribución thalach
ggplot(diseases,aes(x = thalach)) + geom_density(bins =30,fill ="#52BE80") + theme_
bw() + theme_classic() +ggtitle("Distribución atributo thalach") + ylab("Cantidad"
)
```

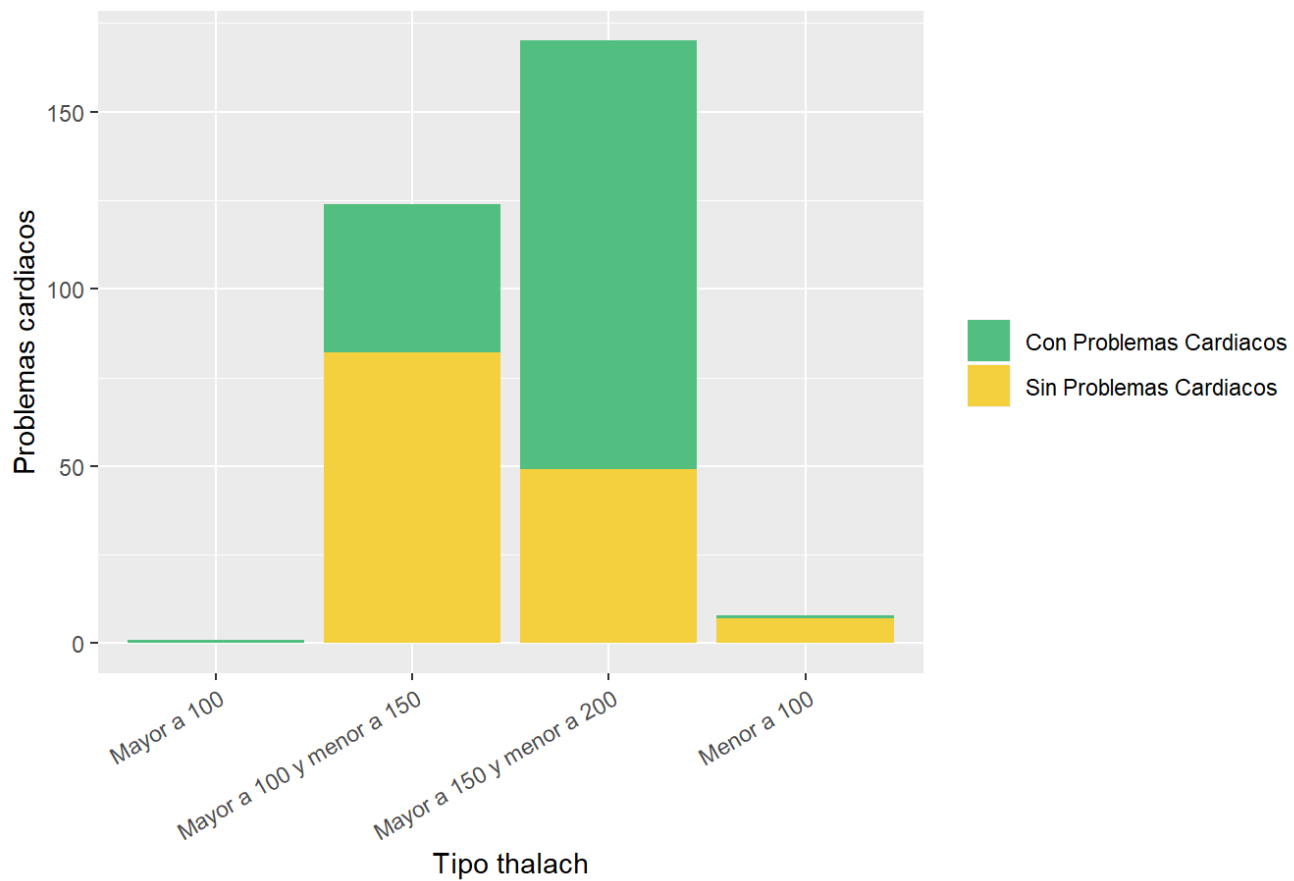
```
## Warning: Ignoring unknown parameters: bins
```

Distribución atributo thalach



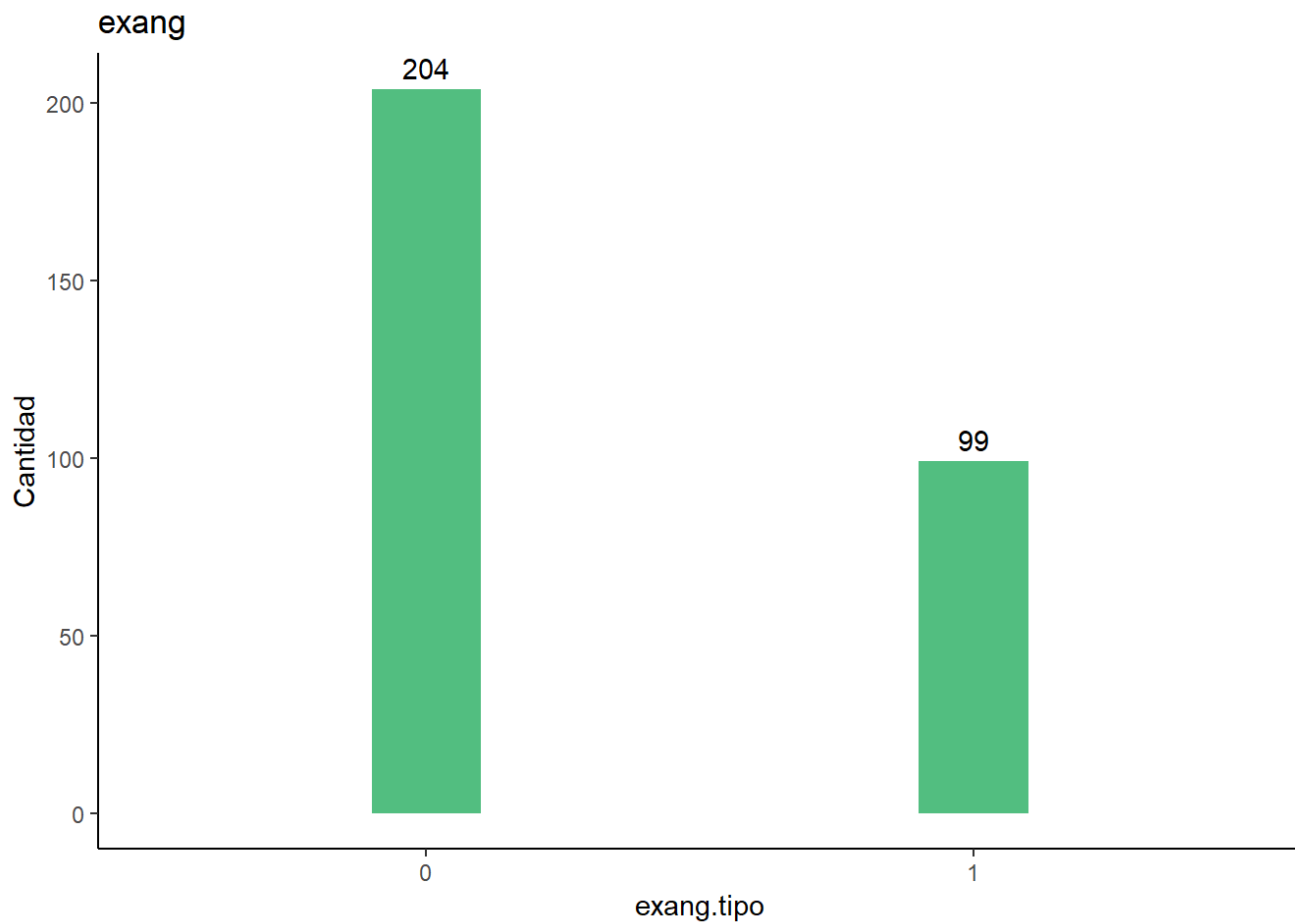
```
# Distribución de problemas cardiacos por thalach
ggplot(diseases,aes(thalach.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo thalach", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardiacos por thalach")+ scale_fill_manual(values=c("#52BE80","#F4D03F")) + theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

Problemas cardiacos por thalach

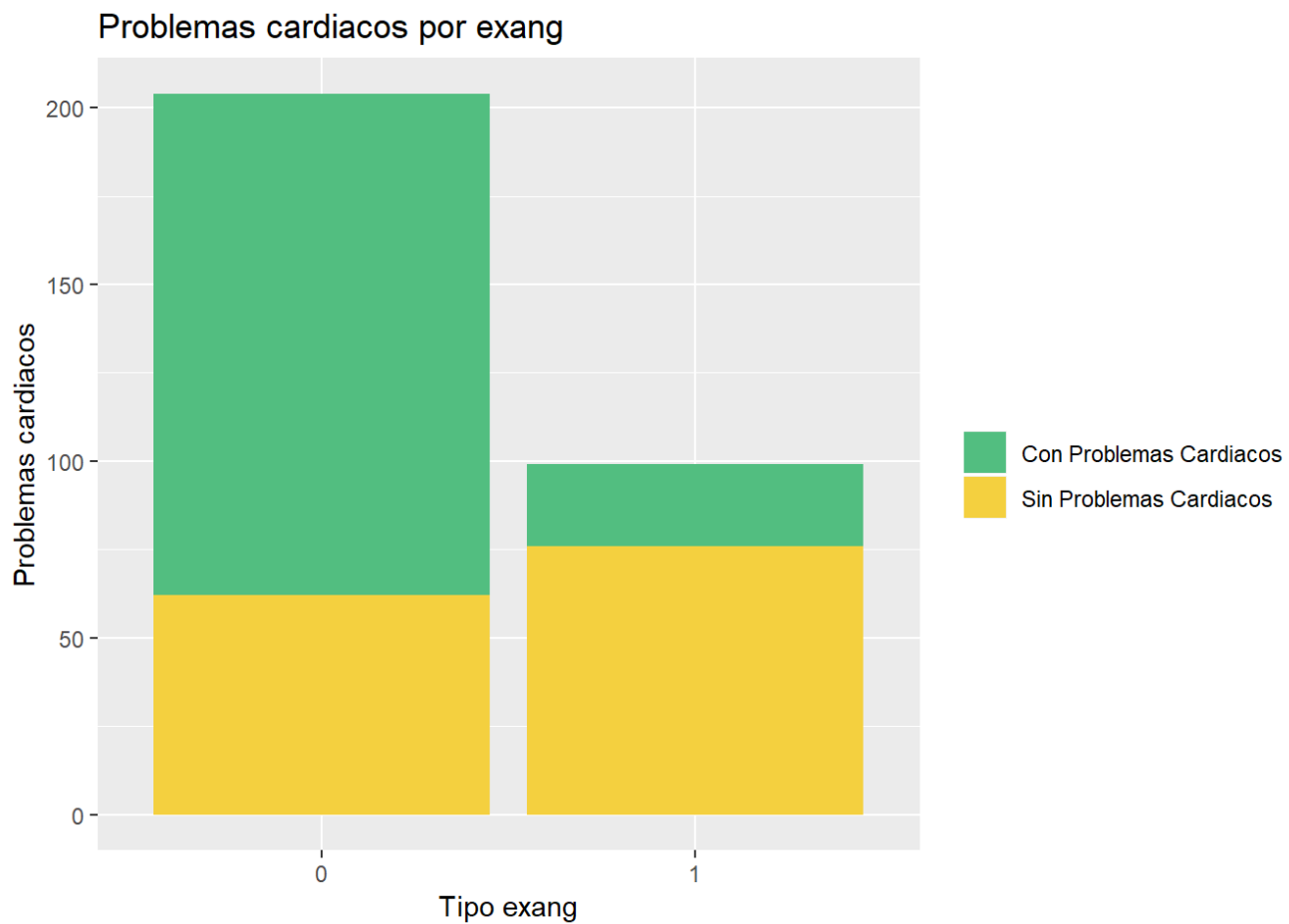


Distribución exang

```
ggplot(diseases,aes(x =exang.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic()  
( ) +ylab("Cantidad") + ggtitle("exang")
```

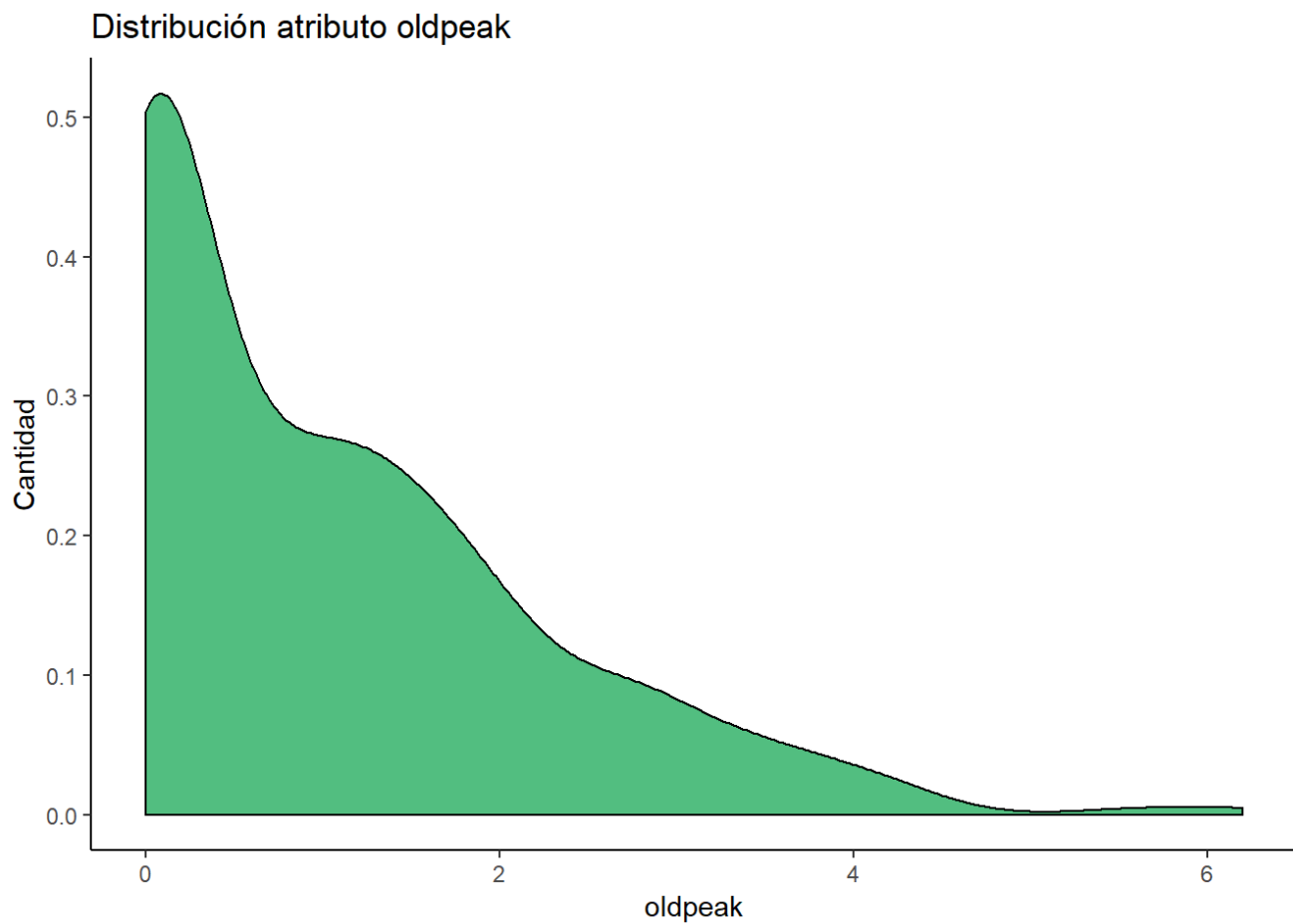


```
# Distribución de problemas cardiacos por exang
ggplot(diseases,aes(exang.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo exang",
y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas
cardiacos por exang")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```



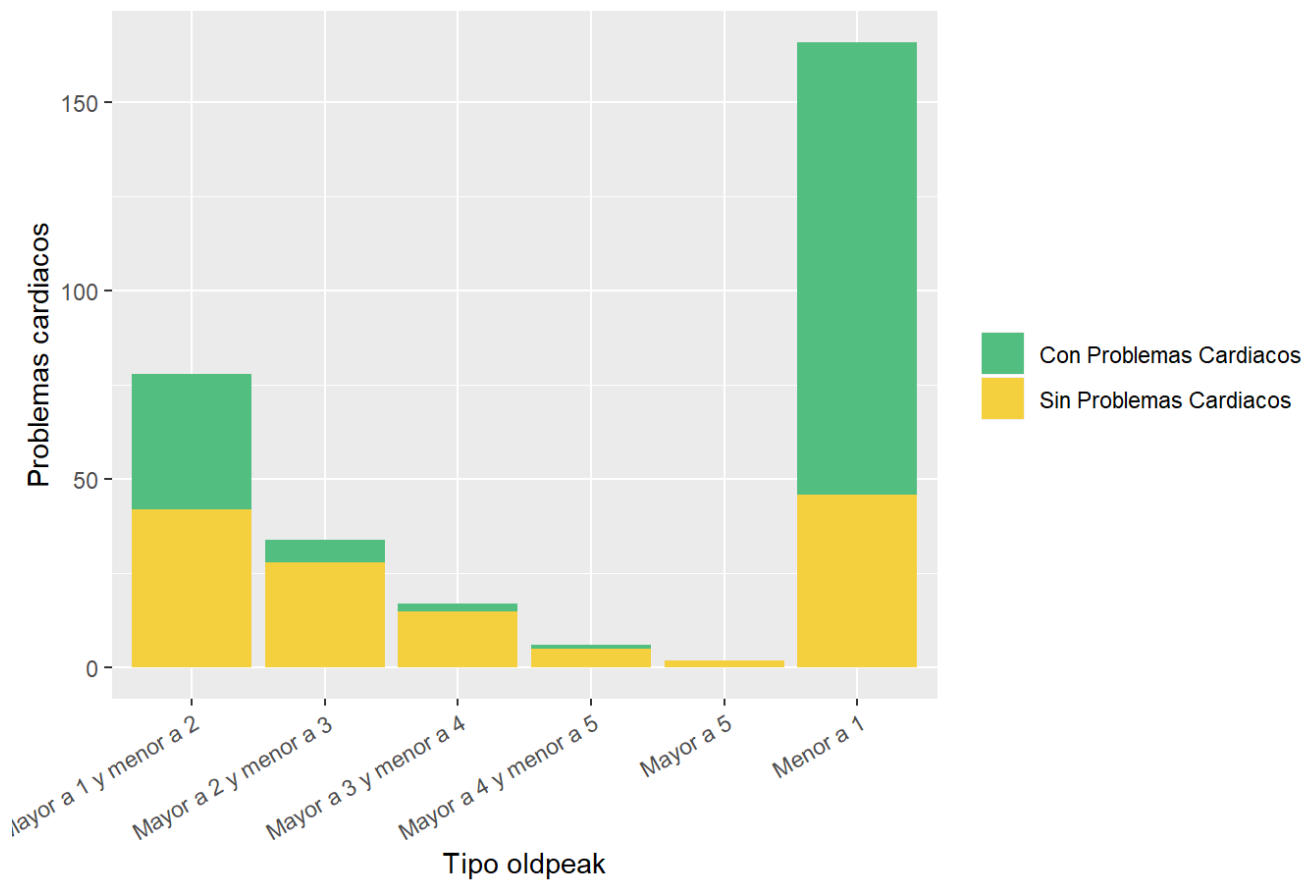
```
# Distribución oldpeak
ggplot(diseases,aes(x = oldpeak)) + geom_density(bins =30,fill ="#52BE80") + theme_
bw() + theme_classic() +ggtitle("Distribución atributo oldpeak") + ylab("Cantidad"
)
```

```
## Warning: Ignoring unknown parameters: bins
```



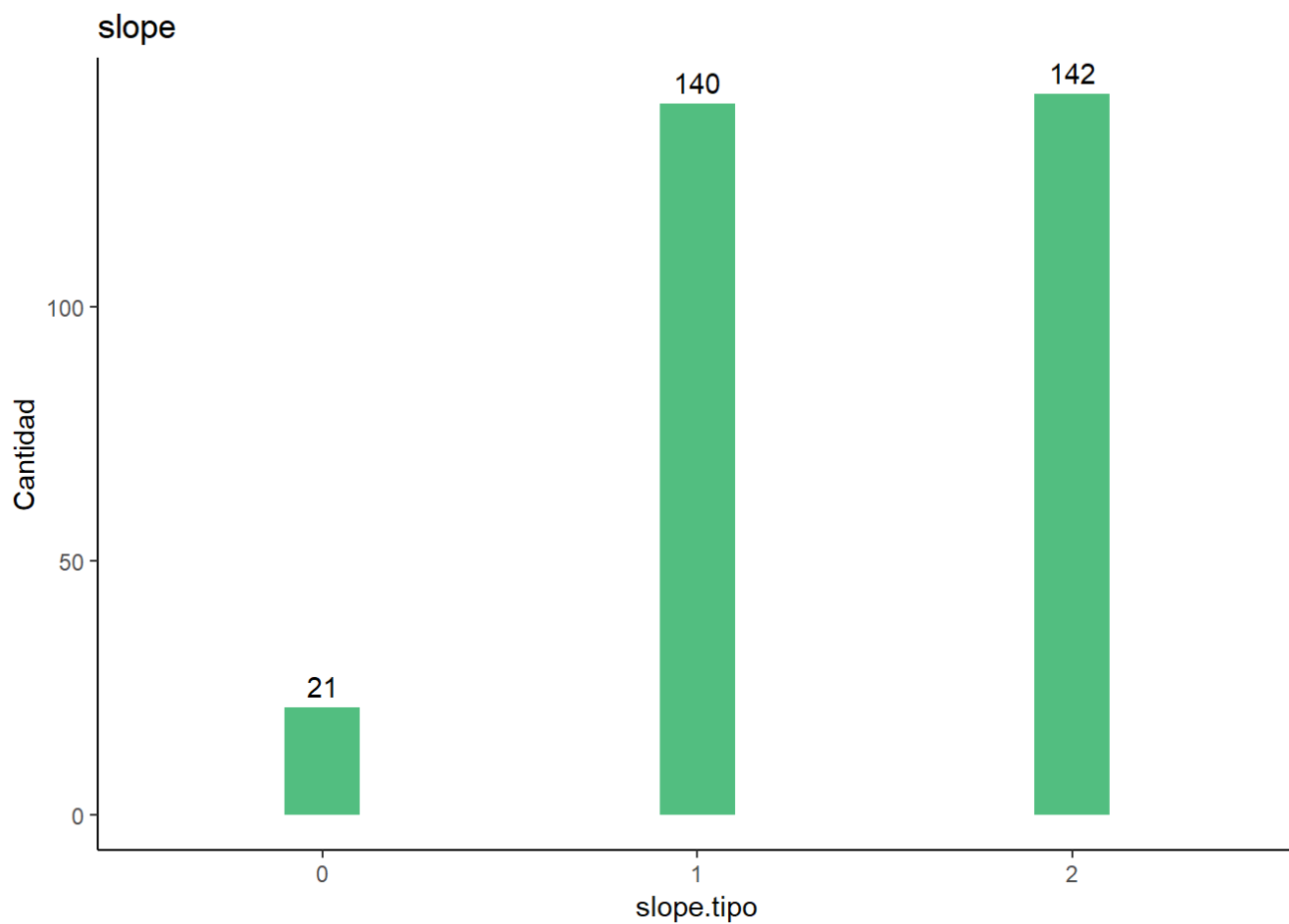
```
# Distribución de problemas cardiacos por oldpeak
ggplot(diseases,aes(oldpeak.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo oldpeak", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardiacos por oldpeak")+ scale_fill_manual(values=c("#52BE80","#F4D03F")) + theme(axis.text.x = element_text(angle = 30, hjust = 1))
```


Problemas cardiacos por oldpeak



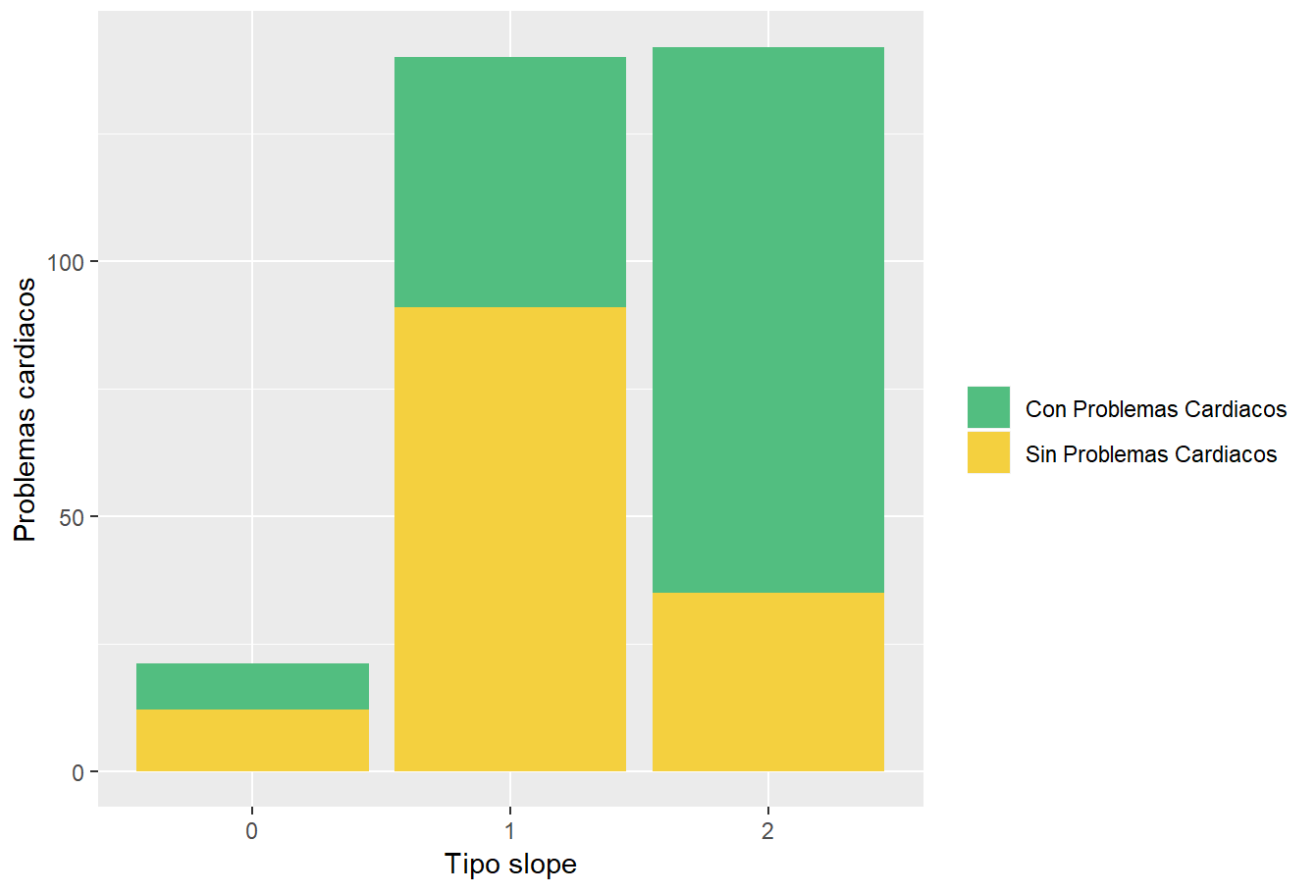
Distribución slope

```
ggplot(diseases,aes(x =slope.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic() +ylab("Cantidad") + ggtitle("slope")
```



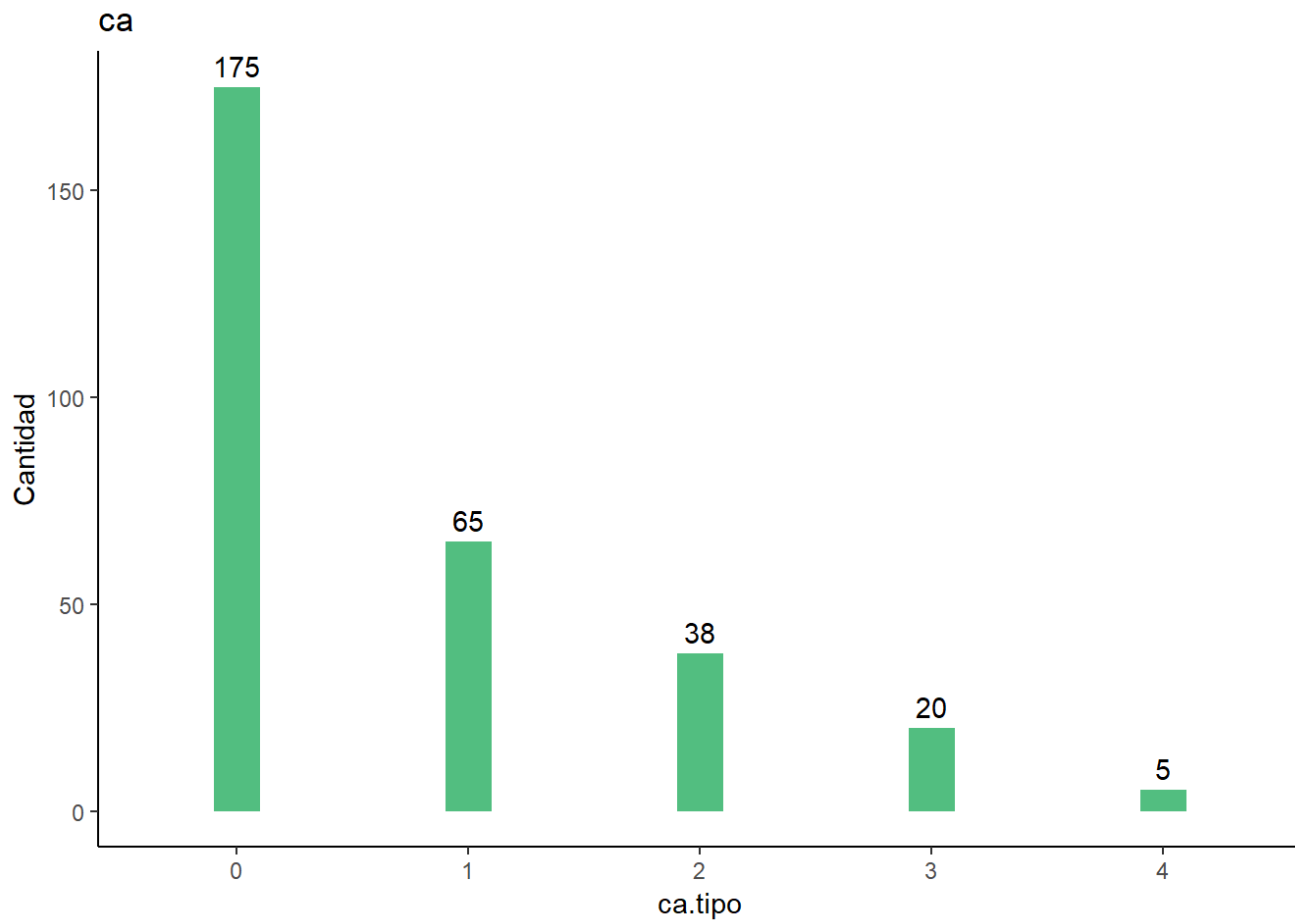
```
# Distribución de problemas cardiacos por slope
ggplot(diseases,aes(slope.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo slope",
y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas
cardiacos por slope")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```

Problemas cardiacos por slope

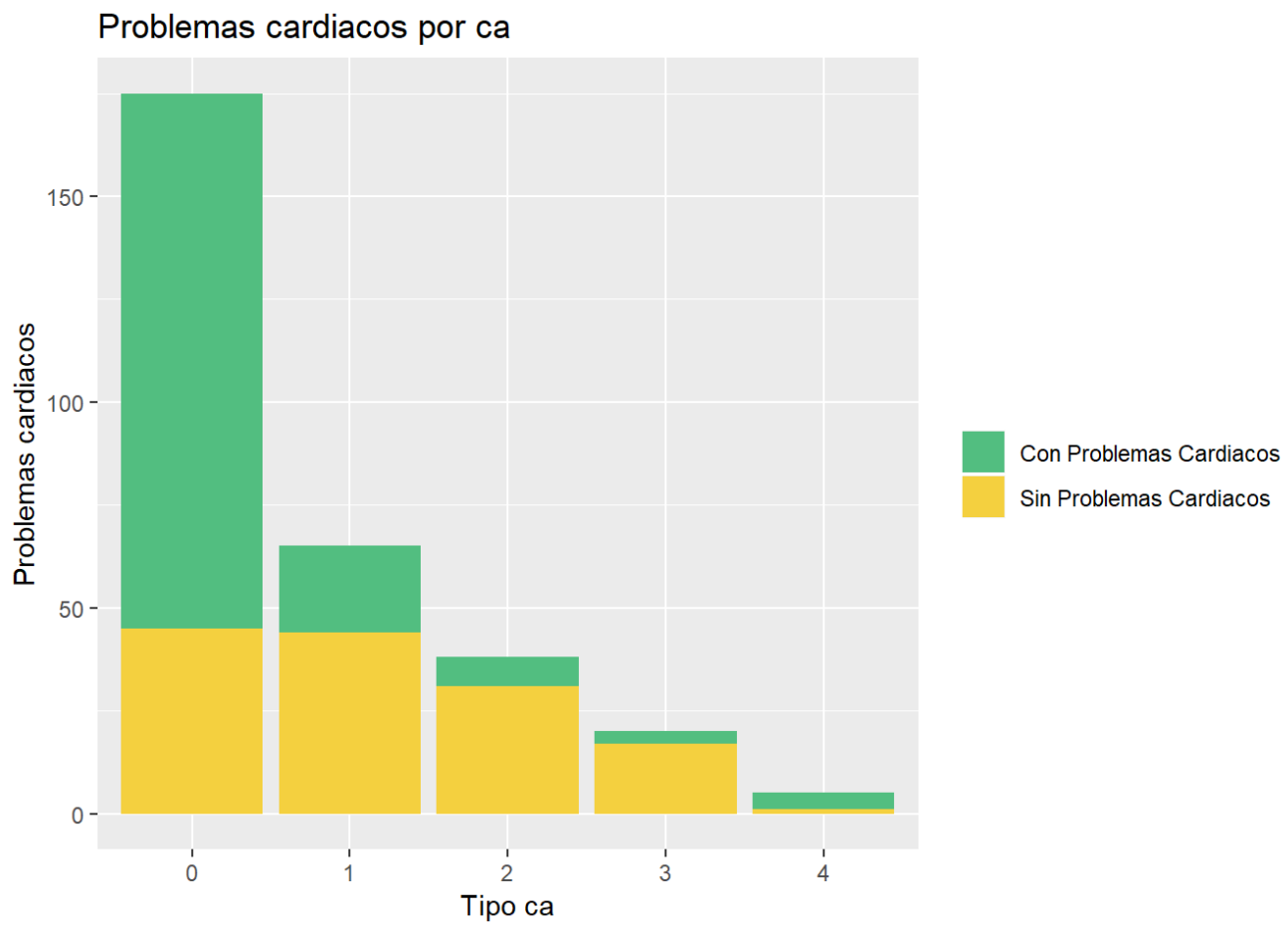


Distribución ca

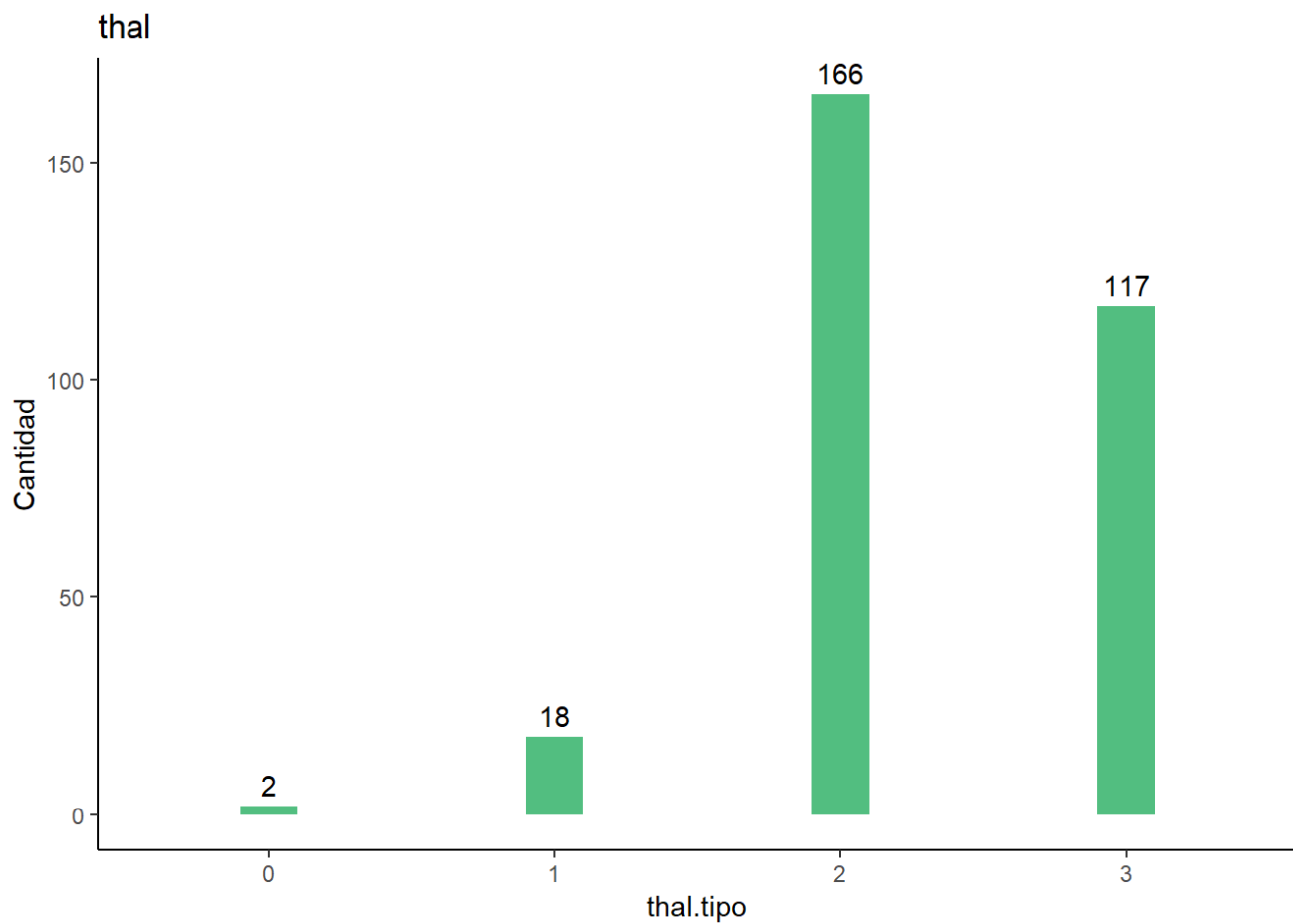
```
ggplot(diseases,aes(x =ca.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_text(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic() +ylab("Cantidad") + ggtitle("ca")
```



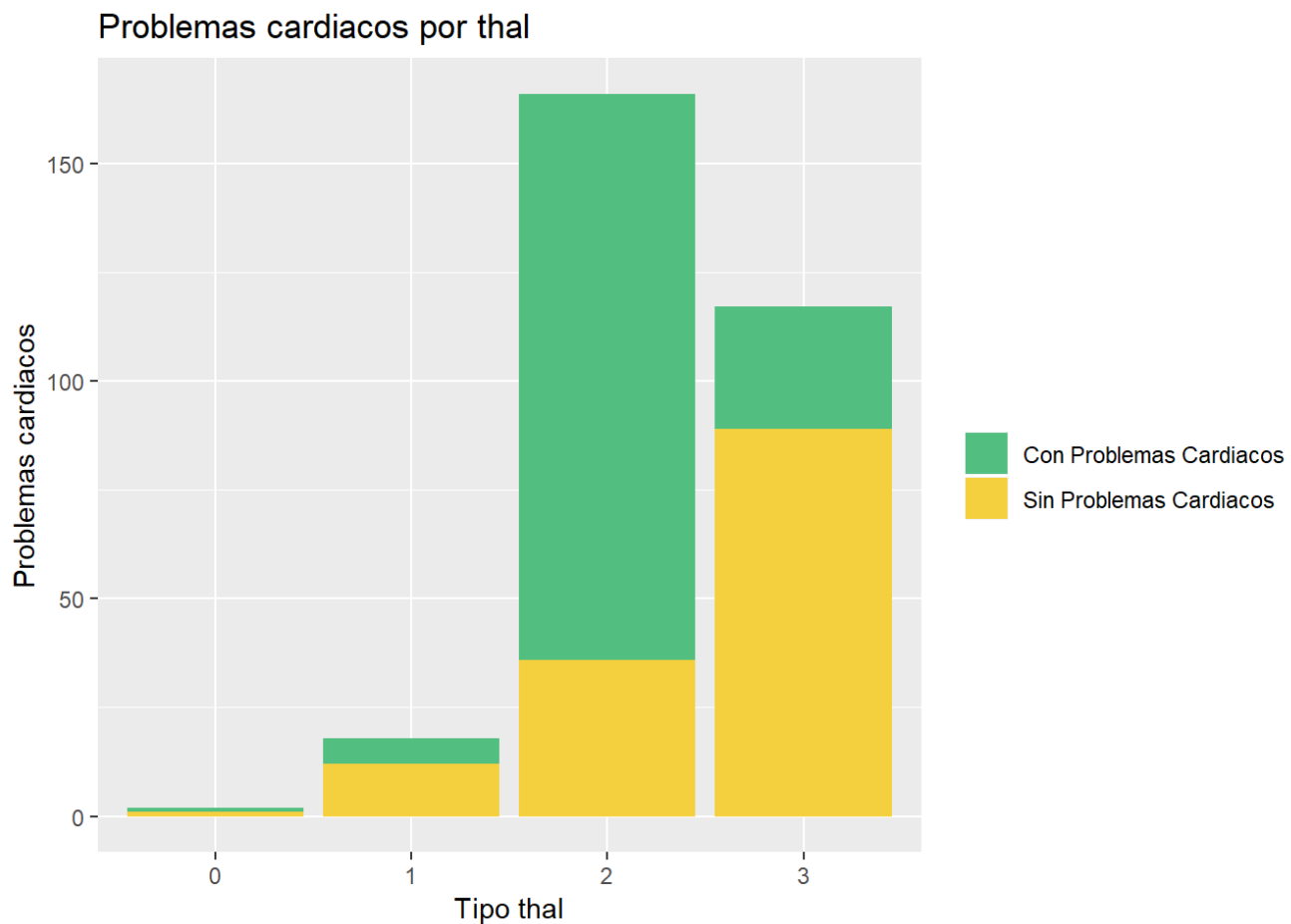
```
# Distribución de problemas cardiacos por ca
# Podemos observar que con valores en cero mayor es la probabilidad de tener problemas cardiacos
ggplot(diseases,aes(ca.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo ca", y="Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas cardiacos por ca")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```



```
# Distribución thal
ggplot(diseases,aes(x =thal.tipo)) + geom_bar(width = 0.2,fill ="#52BE80") + geom_t
ext(stat = 'count',aes(label =..count..),vjust =-0.5) + theme_bw() + theme_classic
() +ylab("Cantidad") + ggtitle("thal")
```



```
# Distribución de problemas cardiacos por thal
ggplot(diseases,aes(thal.tipo,fill=target.tipo))+geom_bar() +labs(x="Tipo thal", y=
"Problemas cardiacos")+ guides(fill=guide_legend(title=""))+ ggtitle("Problemas car
dicos por thal")+ scale_fill_manual(values=c("#52BE80","#F4D03F"))
```



La variable “thal” presenta inconsistencia en su definición versua los datos descritos anteriormente, por lo que no se utilizará en el modelo.

```
diseases$thal = NULL  
diseases$thal.tipo = NULL
```

Como podemos observar, el resto de las variables pueden tener relación en el diagnostico de un problema cardiaco y pueden ser combinadas entre ellas en modelos predictivo.

1.3 Limpieza de los datos.

1.3.1 ¿Los datos contienen ceros o elementos vacíos? ¿Cómo gestionarías cada uno de estos casos?

Primero revisaremos si existen valores NA en el data set, si existiesen les realizaremos el tratamiento correspondiente.

```
# Con la sentencia complete.cases revisamos que cada fila este completa devolviendo  
TRUE, si alguna fila tuviese NA devolvería FALSE. Guardaremos los valores en un ve  
ctor para luego revisarlo.  
vectorNA <-complete.cases(diseases)  
# Revisamos si existe alguna fila con algún valor en NA  
vectorNA[FALSE]
```

```
## logical(0)
```

```
# Otra forma de revisar la existencia de NA es la siguiente.  
sapply(diseases, function(x) sum(is.na(x)))
```

```
##          age          sex          cp          trestbps          chol  
##          0          0          0          0          0  
##          fbs          restecg          thalach          exang          oldpeak  
##          0          0          0          0          0  
##          slope          ca          target          target.tipo          age.tipo  
##          0          0          0          0          0  
##          sex.tipo          cp.tipo          trestbps.tipo          chol.tipo          fbs.tipo  
##          0          0          0          0          0  
##          restecg.tipo          thalach.tipo          exang.tipo          oldpeak.tipo          slope.tipo  
##          0          0          0          0          0  
##          ca.tipo  
##          0
```

Como podemos ver, no existe ninguna fila con valores vacíos. Si hubiesen existido podríamos gestionarlos, dependiendo el caso, eliminando el registro o ingresando alguna media de tendencia central.

```
# Si hubiesemos necesitado eliminar los registros que alguna columna tuviese NA, se  
ría con la siguiente instrucción  
diseases <- na.omit(diseases)
```

```
# Si alguna columna tuviese NA y su distribución de datos fuese uniforme, utilizari  
amos la media para reemplazar el datos faltante.  
# Por ejemplo, veremos el caso para la columna trestbps (datos de la presión arteri  
al del paciente en reposo), en donde, con la siguiente instrucción reemplazaríamos  
los NA con la media  
mean(diseases$age[!is.na(diseases$age)])
```

```
## [1] 54.36634
```

```
diseases$age[is.na(diseases$age)] <- mean(diseases$age[!is.na(diseases$age)])
```

```
# Si alguna columna tuviese NA y su distribución de datos estuviese sesgada, utiliz  
ariamos la mediana para reemplazar el datos faltante.  
# Por ejemplo, veremos el caso para la columna trestbps (datos de la presión arteri  
al del paciente en reposo), en donde, con la siguiente instrucción reemplazaríamos  
los NA con la mediana  
median(diseases$trestbps[!is.na(diseases$trestbps)])
```

```
## [1] 130
```



```
diseases$trestbps[is.na(diseases$trestbps)] <- median(diseases$trestbps[!is.na(diseases$trestbps)])
```

Como podemos observar en el punto 2, al momento de analizar los datos de interés, se mostró la distribución de cada una de las columnas, en donde se logra validar que no existe valores en cero los cuales no aporten la realidad del dato. Por ejemplo, para la variable sex, el valor en cero tiene el significado para las personas de sexo femenino, en cambio si hubiese existido para la variable thalach (maximo ritmo cardiaco alcanzado) algún registro con valor en cero, no hubiese tenido sentido ese valor, el cual hubiesemos tenido que reemplazarlos por algún valor a convenir. Muchas veces tiene más sentido cambiar ese valor cero en NA, ya que en cero nos puede arruinar alguna medida como puede ser la media, en cambio con NA no la podemos considerar para el cálculo de la media.

```
# Para transformar los valores cero en NA, utilizamos la siguiente instrucción  
diseases$thalach[diseases$thalach == 0] <- NA
```

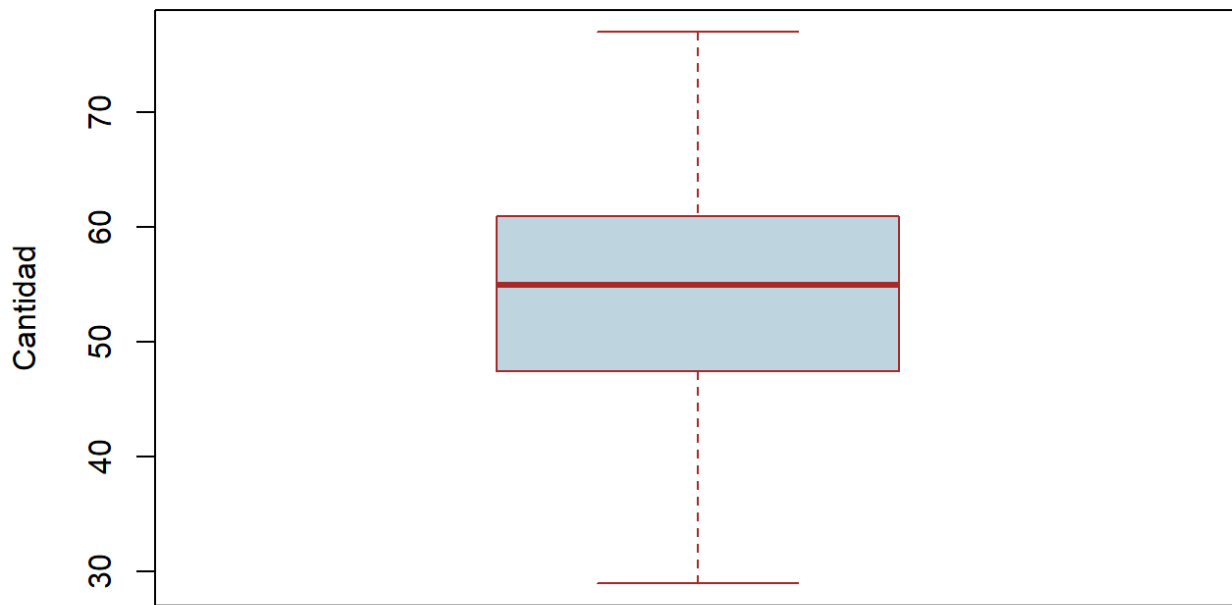
Después de transformarla en NA, podemos reemplazarla por alguna medida de tendencia central con alguna técnica descrita anteriormente.

1.3.2 Identificación y tratamiento de valores extremos.

A continuación identificaremos los valores extremos (outliers) de aquellas variables que sus datos puedan ser atípicos en relación con el resto.

```
# Variable AGE, no se visualiza valores extremos  
boxplot(diseases$age, main = "Distribución variable AGE", ylab = "Cantidad", col = "#bed4de", border = "brown")
```

Distribución variable AGE

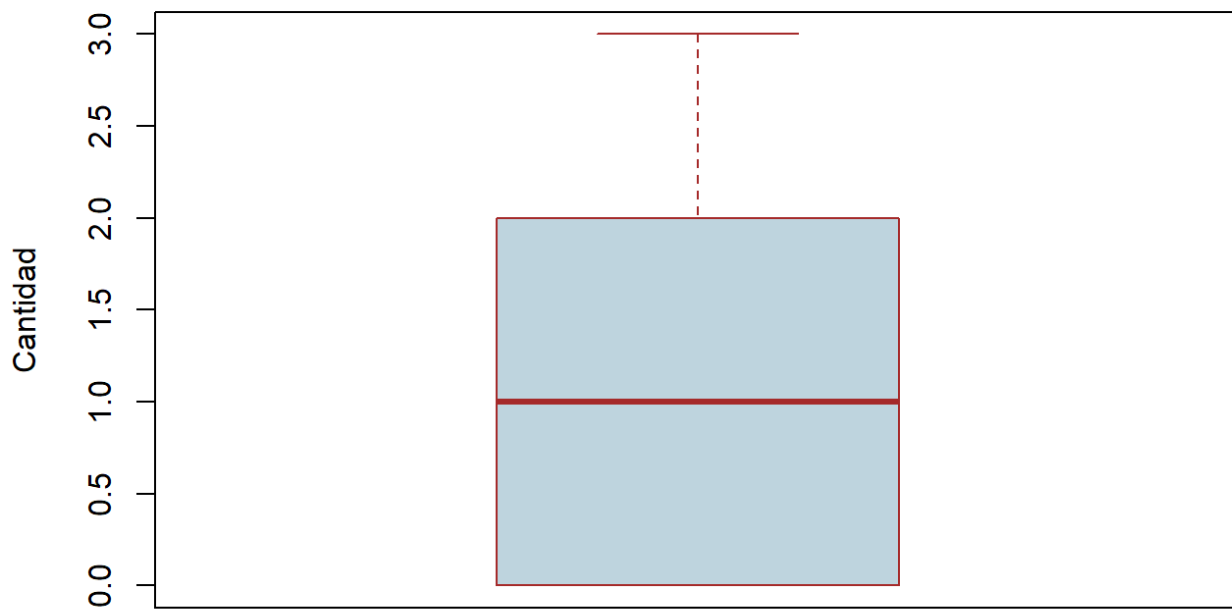


```
boxplot.stats(diseases$age)$out #Muestra los valores extremos si hubiesen
```

```
## numeric(0)
```

```
# Variable CP, no se visualiza valores extremos  
boxplot(diseases$cp, main = "Distribución variable CP", ylab = "Cantidad", col = "#b  
ed4de", border = "brown")
```

Distribución variable CP

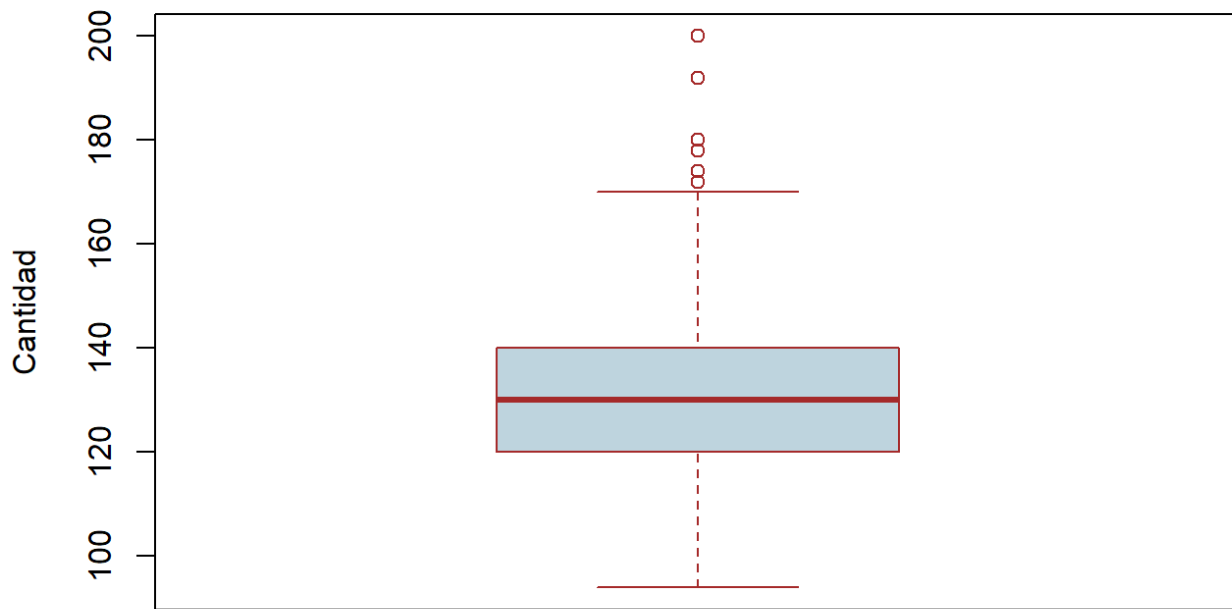


```
boxplot.stats(diseases$cp)$out #Muestra los valores extremos si hubiesen
```

```
## integer(0)
```

```
# Variable trestbps. Podemos ver que existen valores extremos.  
boxplot(diseases$trestbps, main = "Distribución variable Trestbps", ylab = "Cantida  
d", col = "#bed4de", border = "brown")
```

Distribución variable Trestbps



```
# Aquí podemos ver valores que salen de lo común, a continuación los imprimiremos a  
# aquellos valores
```

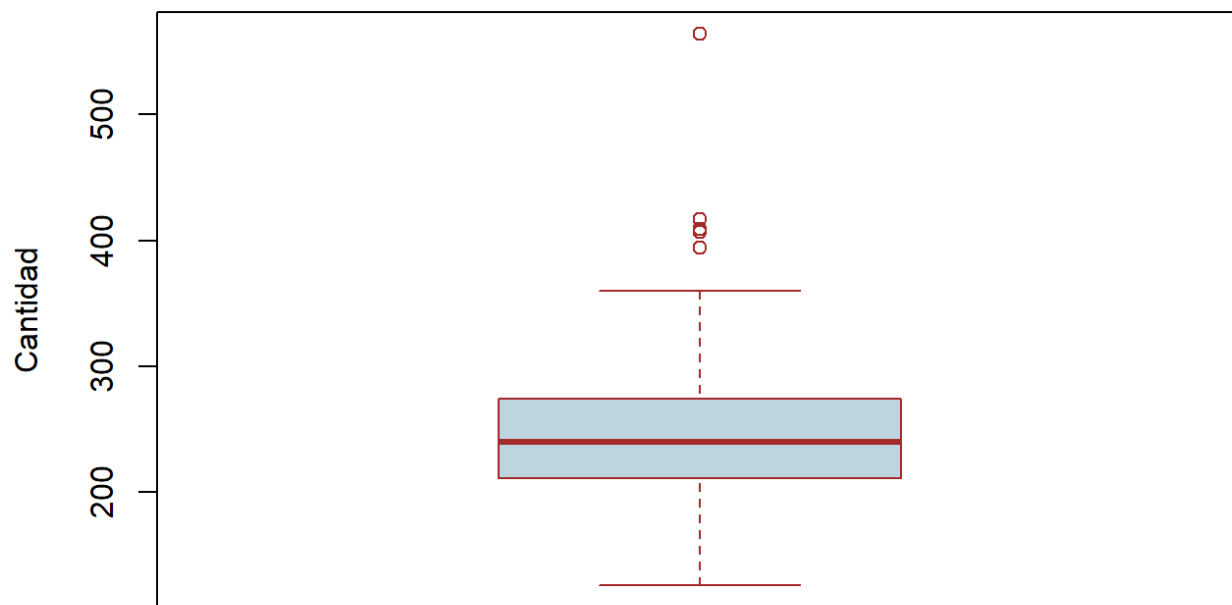
```
boxplot.stats(diseases$trestbps)$out #Muestra los valores extremos si hubiesen
```

```
## [1] 172 178 180 180 200 174 192 178 180
```

```
# Variable chol. Podemos ver que existen valores extremos.
```

```
boxplot(diseases$chol, main = "Distribución variable Chol", ylab = "Cantidad", col  
= "#bed4de", border = "brown")
```

Distribución variable Chol

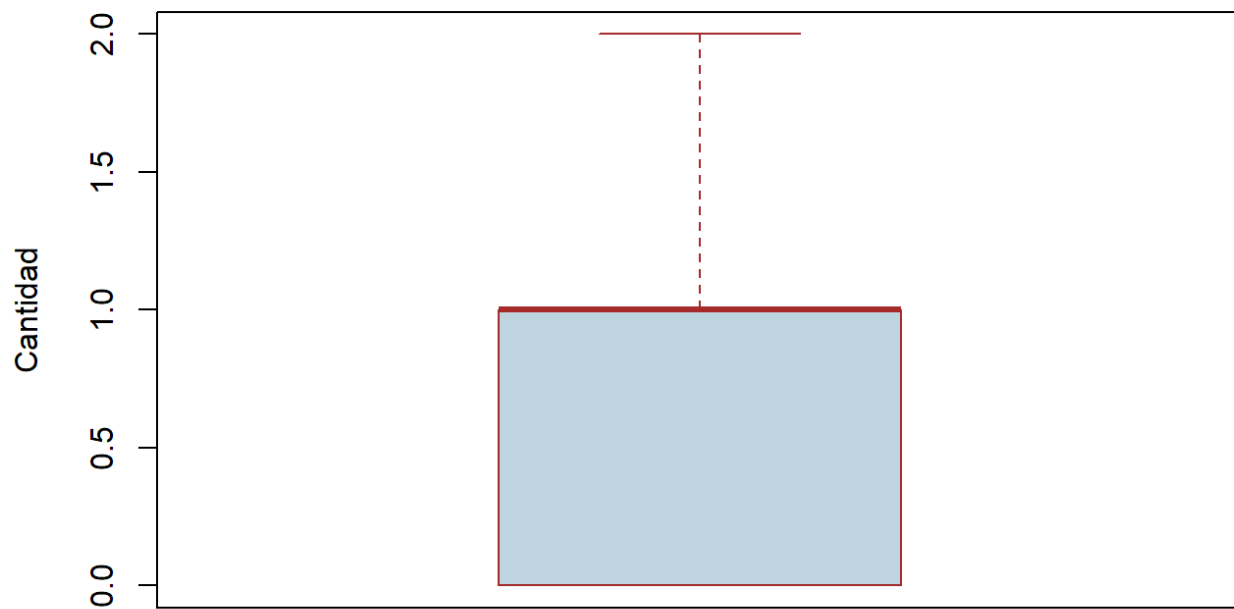


```
# Imprimimos aquellos valores atípicos  
boxplot.stats(diseases$chol)$out #Muestra los valores extremos si hubiesen
```

```
## [1] 417 564 394 407 409
```

```
# Variable restecg, no se visualiza valores extremos  
boxplot(diseases$restecg, main = "Distribución variable Restecg", ylab = "Cantidad",  
col = "#bed4de", border = "brown")
```

Distribución variable Restecg

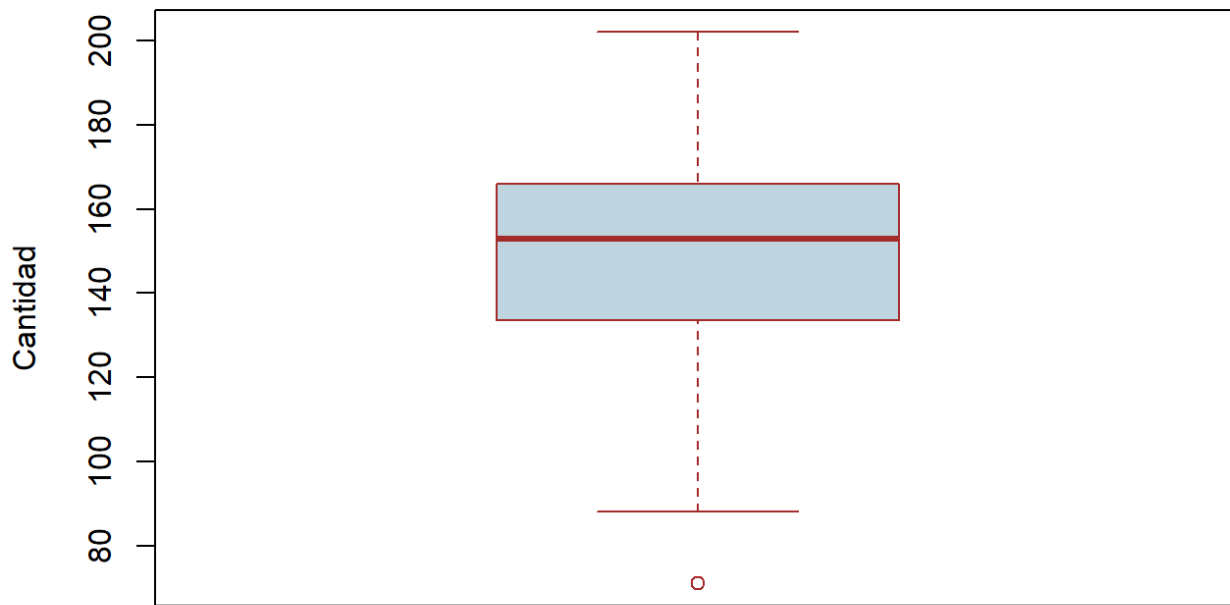


```
boxplot.stats(diseases$restecg)$out #Muestra los valores extremos si hubiesen
```

```
## integer(0)
```

```
# Variable thalach. Podemos ver que existen valores extremos.  
boxplot(diseases$thalach, main = "Distribución variable Thalach", ylab = "Cantidad"  
 , col = "#bed4de", border = "brown")
```

Distribución variable Thalach

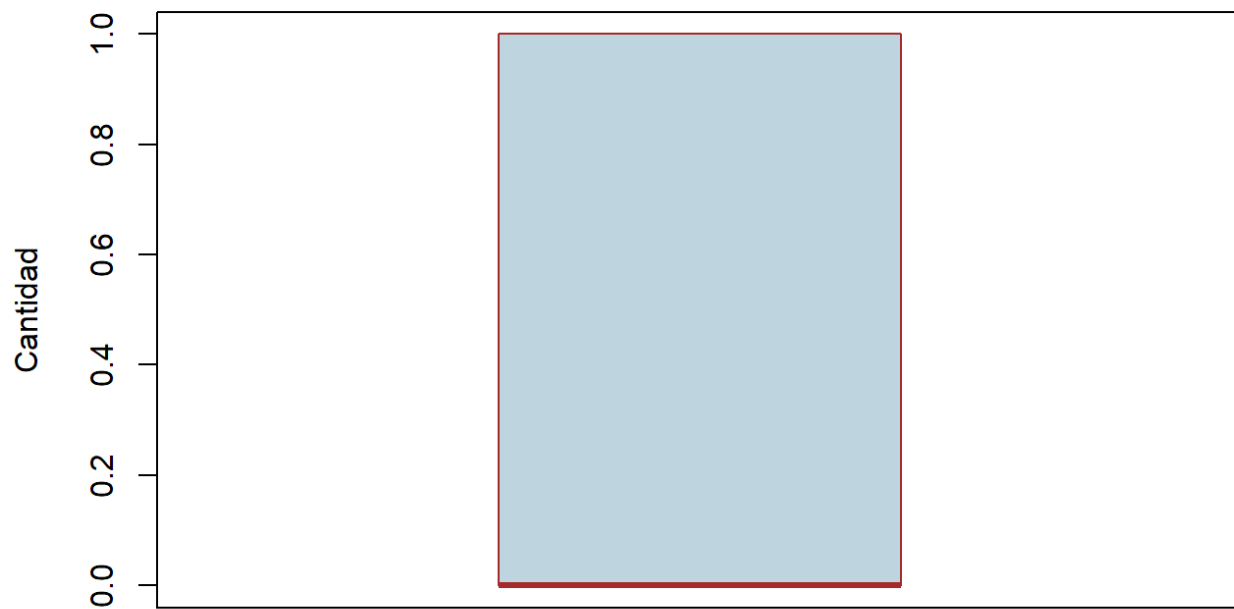


```
# Imprimimos aquellos valores atípicos  
boxplot.stats(diseases$thalach)$out #Muestra los valores extremos si hubiesen
```

```
## [1] 71
```

```
# Variable exang, no se visualiza valores extremos  
boxplot(diseases$exang, main = "Distribución variable Exang", ylab = "Cantidad", col = "#bed4de", border = "brown")
```

Distribución variable Exang

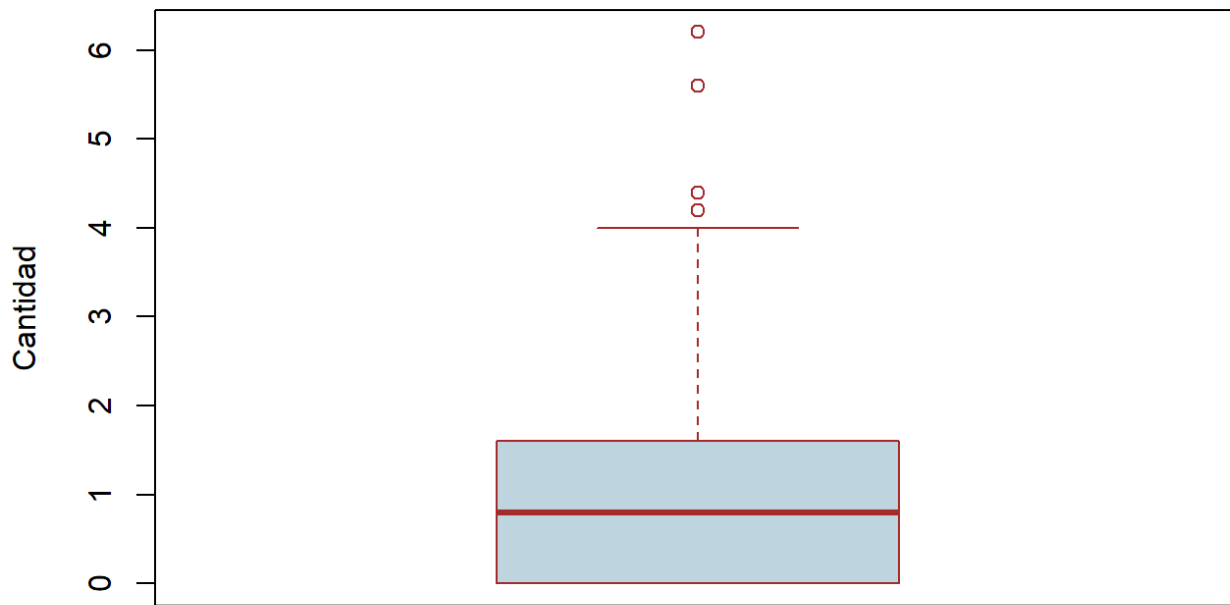


```
boxplot.stats(diseases$exang)$out #Muestra los valores extremos si hubiesen
```

```
## integer(0)
```

```
# Variable oldpeak. Podemos ver que existen valores extremos.  
boxplot(diseases$oldpeak, main = "Distribución variable Oldpeak", ylab = "Cantidad"  
 , col = "#bed4de", border = "brown")
```


Distribución variable Oldpeak

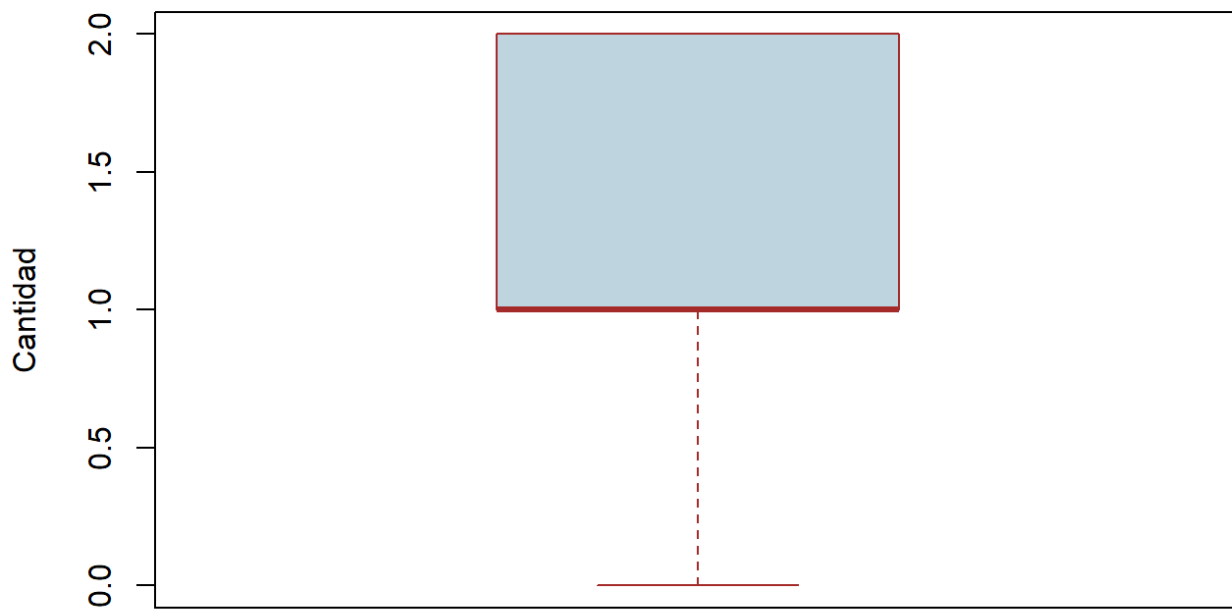


```
# Imprimimos aquellos valores atípicos  
boxplot.stats(diseases$oldpeak)$out #Muestra los valores extremos si hubiesen
```

```
## [1] 4.2 6.2 5.6 4.2 4.4
```

```
# Variable slope, no se visualiza valores extremos  
boxplot(diseases$slope, main = "Distribución variable Slope", ylab = "Cantidad", col = "#bed4de", border = "brown")
```

Distribución variable Slope



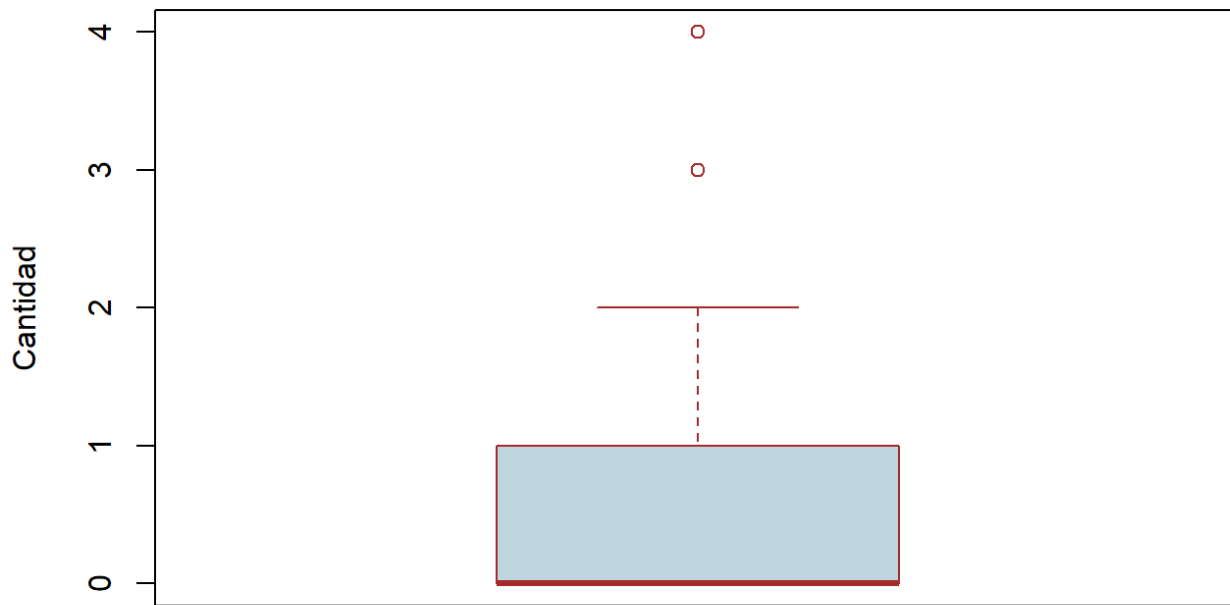
```
boxplot.stats(diseases$slope)$out #Muestra los valores extremos si hubiesen
```

```
## integer(0)
```

```
# Variable ca. Podemos ver que existen valores extremos.
```

```
boxplot(diseases$ca, main = "Distribución variable CA", ylab = "Cantidad", col = "#b  
ed4de", border = "brown")
```

Distribución variable CA



```
# Imprimimos aquellos valores atípicos
boxplot.stats(diseases$ca)$out #Muestra los valores extremos si hubiesen
```

```
## [1] 3 4 3 3 4 4 4 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3
```

Las variables con valores extremos son trestbps, chol, thalach, oldpeak, ca. Estos serán analizados a continuación para ver que realizar con ellos.

- trestbps: Esta variable indica la presión arterial en reposo. Los valores extremos que arroja va desde 172 a 200, con un total de 9 registros. Se revisó la distribución que se describe en el punto 2 y no representan anomalías, ya que se pueden dar estos casos.
- chol: Esta variable indica niveles de colesterol. Los valores que arroja va desde 394 a 564, con un total de 5 registros. Estos valores se escapan considerablemente con el resto de los demás datos. Se eliminarán.
- thalach: Esta variable indica máximos niveles de ritmo cardiaco alcanzado. El valor extremo arrojado es 71, en donde aparece solo una vez en el data set. Este valor es por bajo lo normal, pero dentro de lo esperado para validar si un paciente presenta problemas cardiacos, por lo que no se eliminará.
- oldpeak: Representa la depresión del degmento ST. Los valores que arroja va desde 4.2 a 6.2, con un total de 5 registros. Al revisar estos valores en la distribución de los datos que fueron representados en el punto 2, se escapan del resto de los demás datos, además estos datos extremos se encuentran sesgados en la representación de aquellos pacientes sin problemas cardiacos. Se eliminarán del data set.
- ca: Representa el número de vasos principales coloreados por fluoroscopia. Estos valores extremos que encontramos no representa alteraciones en la realidad o valores que nos vaya a distorcionar nuestros modelos predictivos, por lo que no se eliminarán.

```
#Eliminamos los valores extremos.
```

```
diseases = diseases[diseases$oldpeak < min(boxplot.stats(diseases$oldpeak)$out),]  
diseases = diseases[diseases$chol < min(boxplot.stats(diseases$chol)$out),]
```

Finalmente quedamos con 293 registros de 303 iniciales.

1.3.3 Correlación entre variables

Revisaremos la correlación entre todas las variables numéricas, en donde buscaremos correlaciones menores a -0.5 o mayores a 0.5, siendo 1 y -1 correlaciones perfectas y 0 ninguna correlación, de esta manera consideraremos si es suficiente para una posible eliminación de la variable, por considerarla redundante entre ellas.

```
cor(diseases$age, diseases$sex) # correlación entre age y sex
```

```
## [1] -0.07600017
```

```
cor(diseases$age, diseases$cp) # correlación entre age y cp
```

```
## [1] -0.07350791
```

```
cor(diseases$age, diseases$trestbps) # correlación entre age y trestbps
```

```
## [1] 0.2788777
```

```
cor(diseases$age, diseases$chol) # correlación entre age y chol
```

```
## [1] 0.1840307
```

```
cor(diseases$age, diseases$fbs) # correlación entre age y fbs
```

```
## [1] 0.1215168
```

```
cor(diseases$age, diseases$restecg) # correlación entre age y restecg
```

```
## [1] -0.1032196
```

```
cor(diseases$age, diseases$thalach) # correlación entre age y thalach
```

```
## [1] -0.4105411
```

```
cor(diseases$age, diseases$exang) # correlación entre age y exang
```

```
## [1] 0.1144832
```

```
cor(diseases$age, diseases$oldpeak) # correlación entre age y oldpeak
```

```
## [1] 0.2062777
```

```
cor(diseases$age, diseases$slope) # correlación entre age y slope
```

```
## [1] -0.1598506
```

```
cor(diseases$age, diseases$ca) # correlación entre age y ca
```

```
## [1] 0.2813552
```

```
cor(diseases$sex, diseases$cp) # correlación entre sex y cp
```

```
## [1] -0.05873129
```

```
cor(diseases$sex, diseases$trestbps) # correlación entre sex y trestbps
```

```
## [1] -0.0510139
```

```
cor(diseases$sex, diseases$chol) # correlación entre sex y chol
```

```
## [1] -0.1440484
```

```
cor(diseases$sex, diseases$fbs) # correlación entre sex y fbs
```

```
## [1] 0.05209214
```

```
cor(diseases$sex, diseases$restecg) # correlación entre sex y restecg
```

```
## [1] -0.1009395
```

```
cor(diseases$sex, diseases$thalach) # correlación entre sex y thalach
```

```
## [1] -0.03200088
```

```
cor(diseases$sex, diseases$exang) # correlación entre sex y exang
```

```
## [1] 0.1337944
```

```
cor(diseases$sex, diseases$oldpeak) # correlación entre sex y oldpeak
```

```
## [1] 0.1272768
```

```
cor(diseases$sex, diseases$slope) # correlación entre sex y slope
```

```
## [1] -0.03589005
```

```
cor(diseases$sex, diseases$ca) # correlación entre sex y ca
```

```
## [1] 0.1430607
```

```
cor(diseases$cp, diseases$trestbps) # correlación entre cp y trestbps
```

```
## [1] 0.04396564
```

```
cor(diseases$cp, diseases$chol) # correlación entre cp y chol
```

```
## [1] -0.09790354
```

```
cor(diseases$cp, diseases$fbs) # correlación entre cp y fbs
```

```
## [1] 0.08422713
```

```
cor(diseases$cp, diseases$restecg) # correlación entre cp y restecg
```

```
## [1] 0.0622962
```

```
cor(diseases$cp, diseases$thalach) # correlación entre cp y thalach
```

```
## [1] 0.2934913
```

```
cor(diseases$cp, diseases$exang) # correlación entre cp y exang
```

```
## [1] -0.3945565
```

```
cor(diseases$cp, diseases$oldpeak) # correlación entre cp y oldpeak
```

```
## [1] -0.1310823
```

```
cor(diseases$cp, diseases$slope) # correlación entre cp y slope
```

```
## [1] 0.110581
```

```
cor(diseases$cp, diseases$ca) # correlación entre cp y ca
```

```
## [1] -0.1569964
```

```
cor(diseases$trestbps, diseases$chol) # correlación entre trestbps y chol
```

```
## [1] 0.1536378
```

```
cor(diseases$trestbps, diseases$fbs) # correlación entre trestbps y fbs
```

```
## [1] 0.1871295
```

```
cor(diseases$trestbps, diseases$restecg) # correlación entre trestbps y restecg
```

```
## [1] -0.09404663
```

```
cor(diseases$trestbps, diseases$thalach) # correlación entre trestbps y thalach
```

```
## [1] -0.04207942
```

```
cor(diseases$trestbps, diseases$exang) # correlación entre trestbps y exang
```

```
## [1] 0.07529997
```

```
cor(diseases$trestbps, diseases$oldpeak) # correlación entre trestbps y oldpeak
```

```
## [1] 0.1576912
```

```
cor(diseases$trestbps, diseases$slope) # correlación entre trestbps y slope
```

```
## [1] -0.09781491
```

```
cor(diseases$trestbps, diseases$ca) # correlación entre trestbps y ca
```

```
## [1] 0.0964481
```

```
cor(diseases$chol, diseases$fbs) # correlación entre chol y fbs
```

```
## [1] 0.01127574
```

```
cor(diseases$chol, diseases$restecg) # correlación entre chol y restecg
```

```
## [1] -0.1290889
```

```
cor(diseases$chol, diseases$thalach) # correlación entre chol y thalach
```

```
## [1] -0.02989011
```

```
cor(diseases$chol, diseases$exang) # correlación entre chol y exang
```

```
## [1] 0.1029059
```

```
cor(diseases$chol, diseases$oldpeak) # correlación entre chol y oldpeak
```

```
## [1] 0.01478626
```

```
cor(diseases$chol, diseases$slope) # correlación entre chol y slope
```

```
## [1] 0.02482699
```

```
cor(diseases$chol, diseases$ca) # correlación entre chol y ca
```

```
## [1] 0.05553339
```



```
cor(diseases$fbs, diseases$restecg) # correlación entre fbs y restecg
```

```
## [1] -0.08147533
```

```
cor(diseases$fbs, diseases$thalach) # correlación entre fbs y thalach
```

```
## [1] -0.01524296
```

```
cor(diseases$fbs, diseases$exang) # correlación entre fbs y exang
```

```
## [1] 0.03223467
```

```
cor(diseases$fbs, diseases$oldpeak) # correlación entre fbs y oldpeak
```

```
## [1] 0.04151467
```

```
cor(diseases$fbs, diseases$slope) # correlación entre fbs y slope
```

```
## [1] -0.0894874
```

```
cor(diseases$fbs, diseases$ca) # correlación entre fbs y ca
```

```
## [1] 0.1539619
```

```
cor(diseases$restecg, diseases$thalach) # correlación entre restecg y thalach
```

```
## [1] 0.06174863
```

```
cor(diseases$restecg, diseases$exang) # correlación entre restecg y exang
```

```
## [1] -0.08563258
```

```
cor(diseases$restecg, diseases$oldpeak) # correlación entre restecg y oldpeak
```

```
## [1] -0.08006405
```

```
cor(diseases$restecg, diseases$slope) # correlación entre restecg y slope
```

```
## [1] 0.1093412
```

```
cor(diseases$restecg, diseases$ca) # correlación entre restecg y ca
```

```
## [1] -0.09022665
```

```
cor(diseases$thalach, diseases$exang) # correlación entre thalach y exang
```

```
## [1] -0.3731303
```

```
cor(diseases$thalach, diseases$oldpeak) # correlación entre thalach y oldpeak
```

```
## [1] -0.3479164
```

```
cor(diseases$thalach, diseases$slope) # correlación entre thalach y slope
```

```
## [1] 0.3848138
```

```
cor(diseases$thalach, diseases$ca) # correlación entre thalach y ca
```

```
## [1] -0.2141315
```

```
cor(diseases$exang, diseases$oldpeak) # correlación entre exang y oldpeak
```

```
## [1] 0.3264294
```

```
cor(diseases$exang, diseases$slope) # correlación entre exang y slope
```

```
## [1] -0.2741821
```

```
cor(diseases$exang, diseases$ca) # correlación entre exang y ca
```

```
## [1] 0.1203961
```

```
cor(diseases$oldpeak, diseases$slope) # correlación entre oldpeak y slope
```

```
## [1] -0.537684
```

```
cor(diseases$oldpeak, diseases$ca) # correlación entre oldpeak y ca
```

```
## [1] 0.1683248
```

```
cor(diseases$slope, diseases$ca) # correlación entre slope y ca
```

```
## [1] -0.05237711
```

Como podemos observar, existe una correlación negativa moderada -0.537684, entre las variables oldpeak y slope, ésta no es muy alta pero se realizarán las pruebas cuando generemos los modelos predictivos, considerando las dos y luego solo una de ellas para ver si mejora el porcentaje de predicción.

```
if(!require(PerformanceAnalytics)){  
  install.packages('PerformanceAnalytics', repos='http://cran.us.r-project.org')  
  library(PerformanceAnalytics)  
}
```

```
## Loading required package: PerformanceAnalytics
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##  
## Attaching package: 'zoo'
```

```
## The following object is masked from 'package:timeSeries':  
##  
##      time<-
```

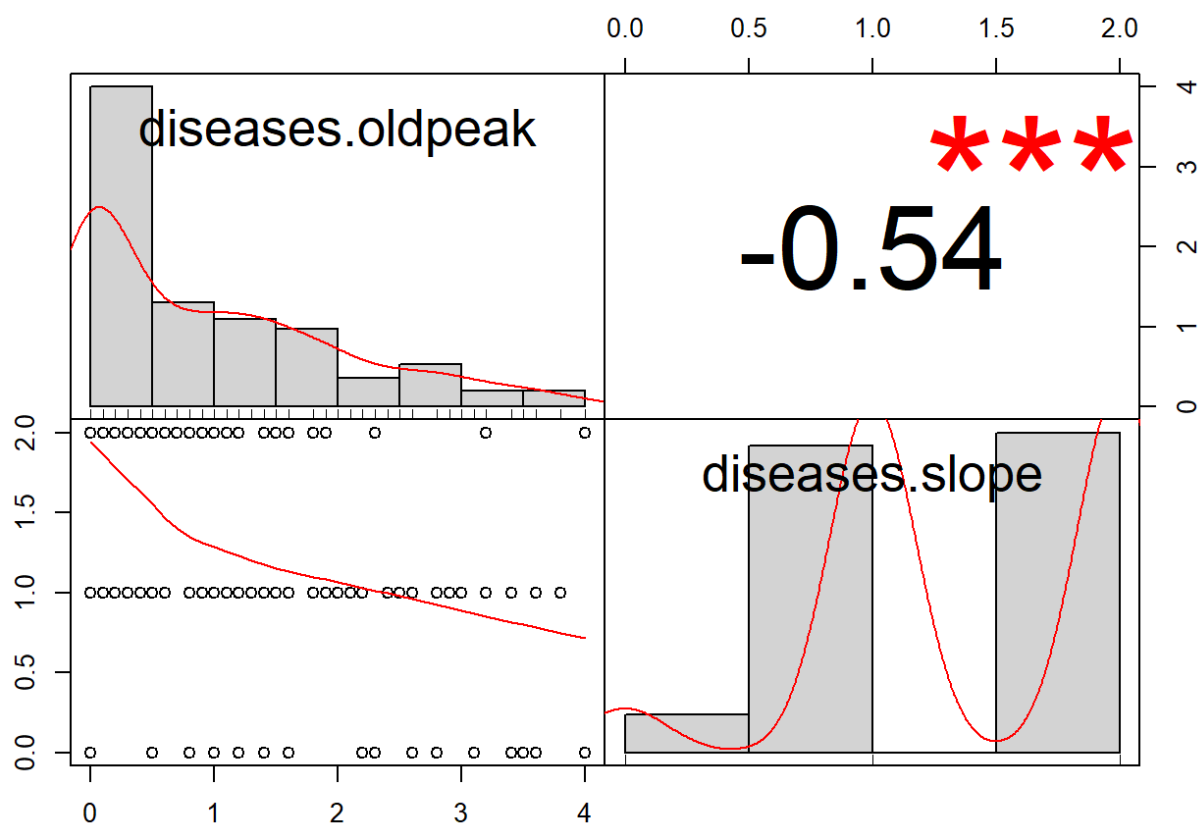
```
## The following objects are masked from 'package:base':  
##  
##      as.Date, as.Date.numeric
```

```
##  
## Attaching package: 'PerformanceAnalytics'
```

```
## The following objects are masked from 'package:timeDate':  
##  
##      kurtosis, skewness
```

```
## The following object is masked from 'package:graphics':
##
##      legend
```

```
dat1 <- data.frame(diseases$oldpeak, diseases$slope)
chart.Correlation(dat1) # graficamos la correlación
```



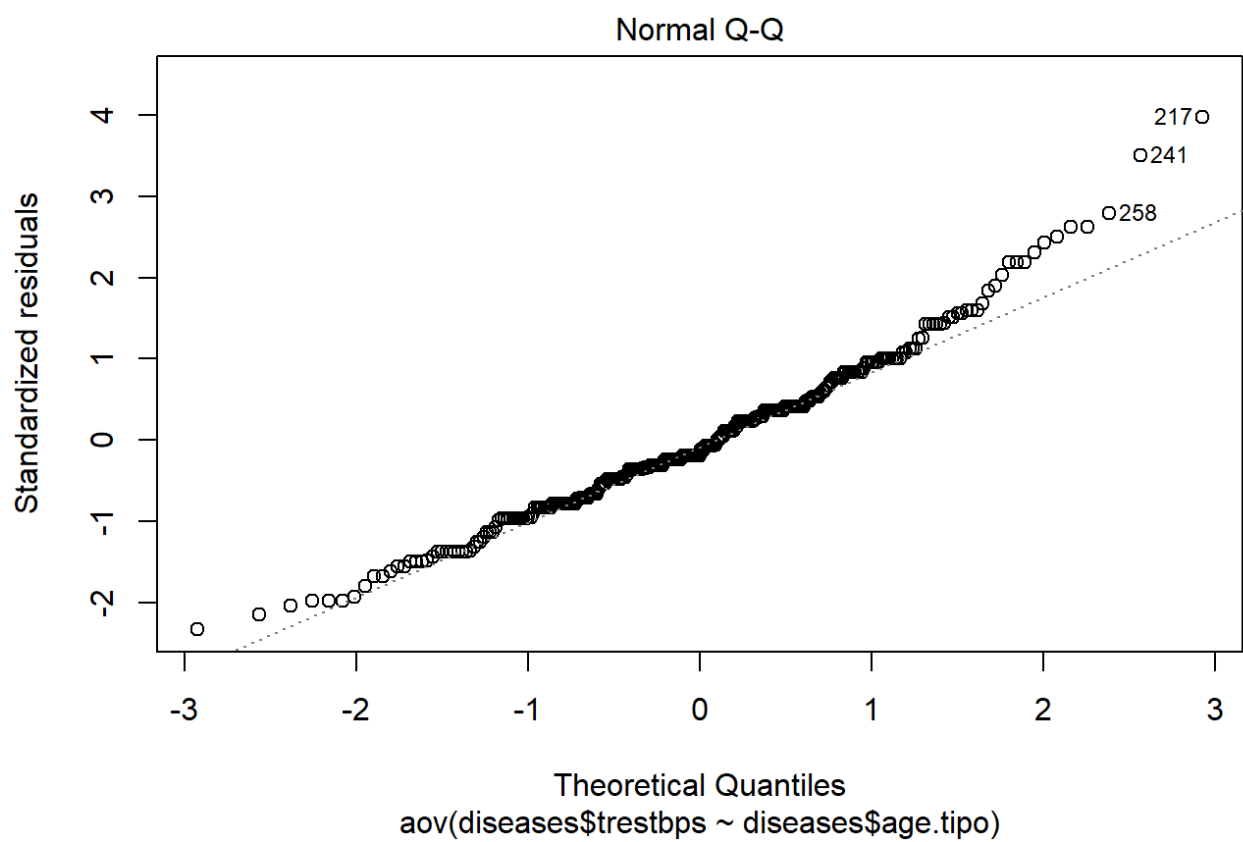
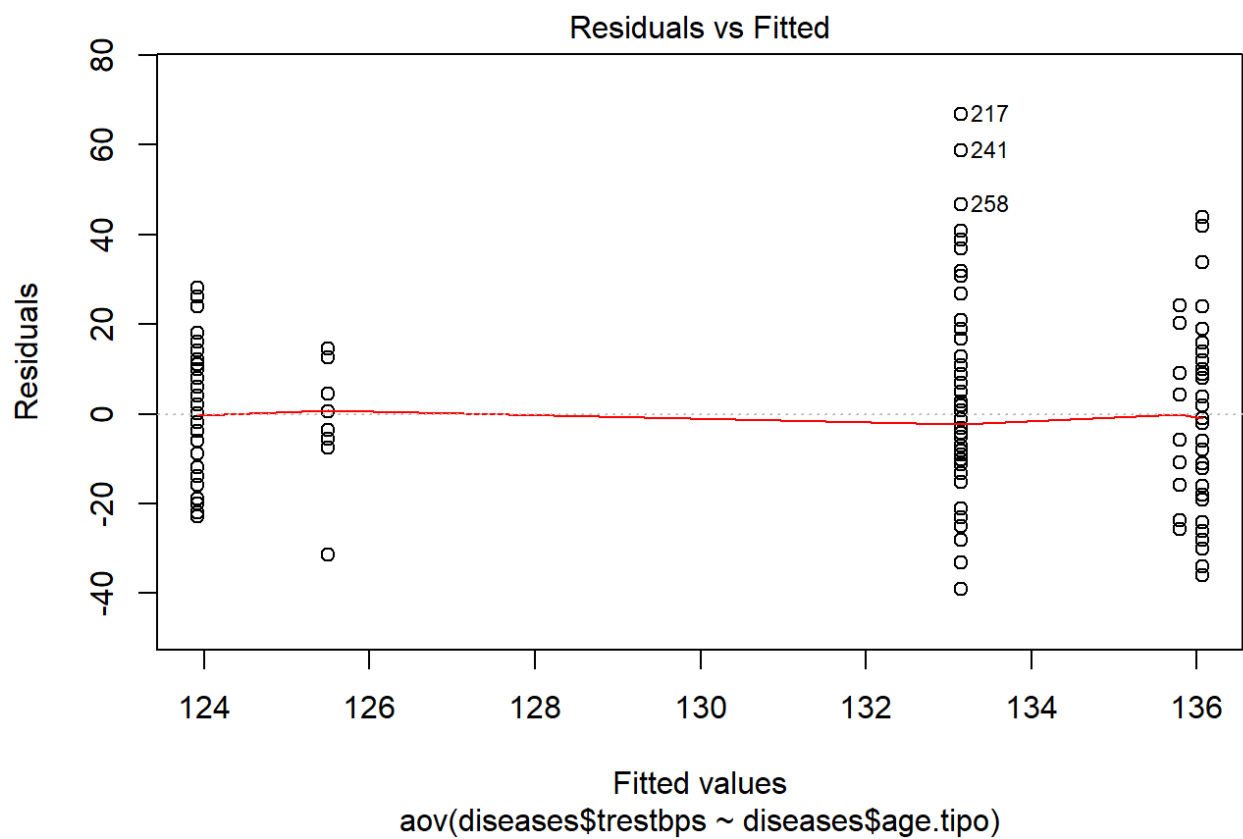
Realizado este estudio, vamos a realizar análisis ANOVA para ver si se encuentran diferencias significativas entre las media del valor de `trestbps` (presión arterial en reposo) y otras variables entre los diferentes grupos de edad. Este estudio nos puede permitir ver las características de las variables y quizá sacar alguna conclusión de prioridad de variables a incluir en algún model que queramos estudiar.

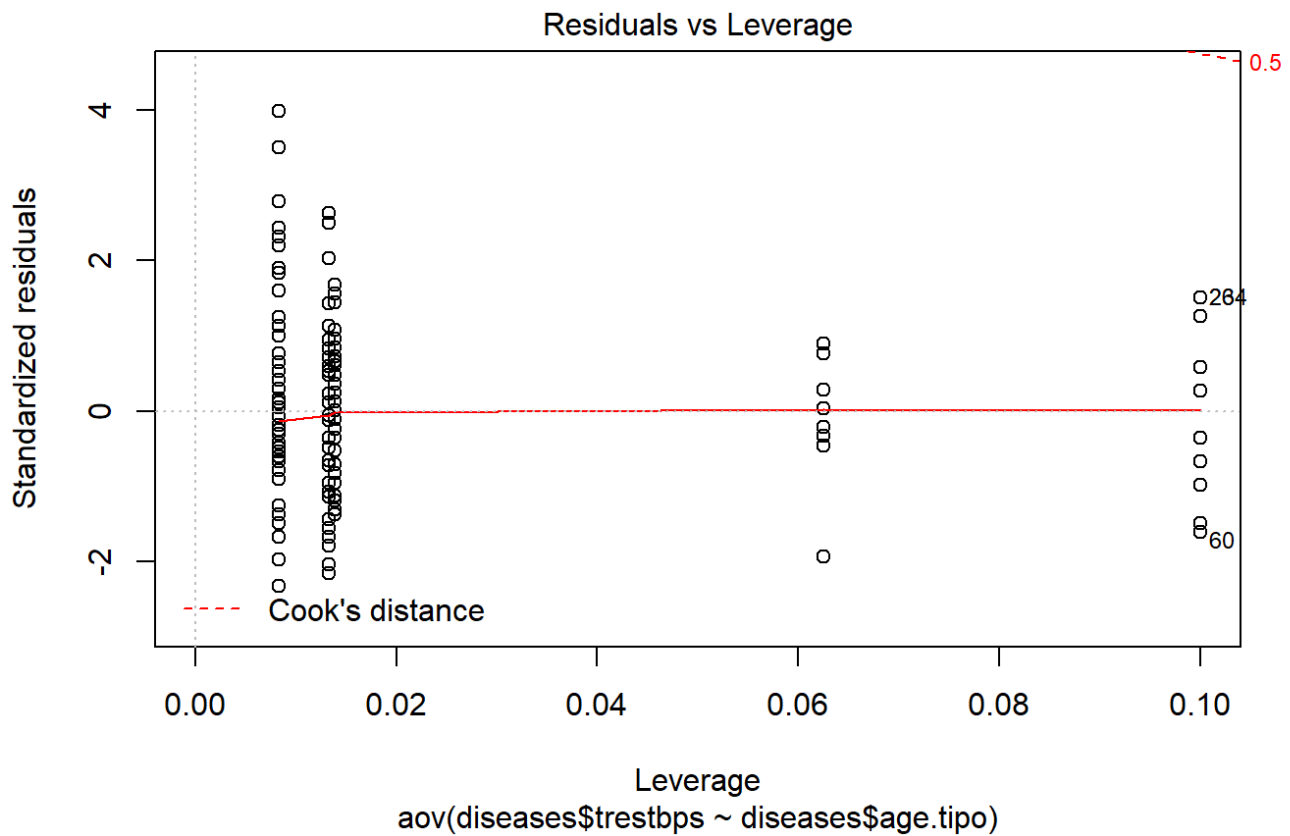
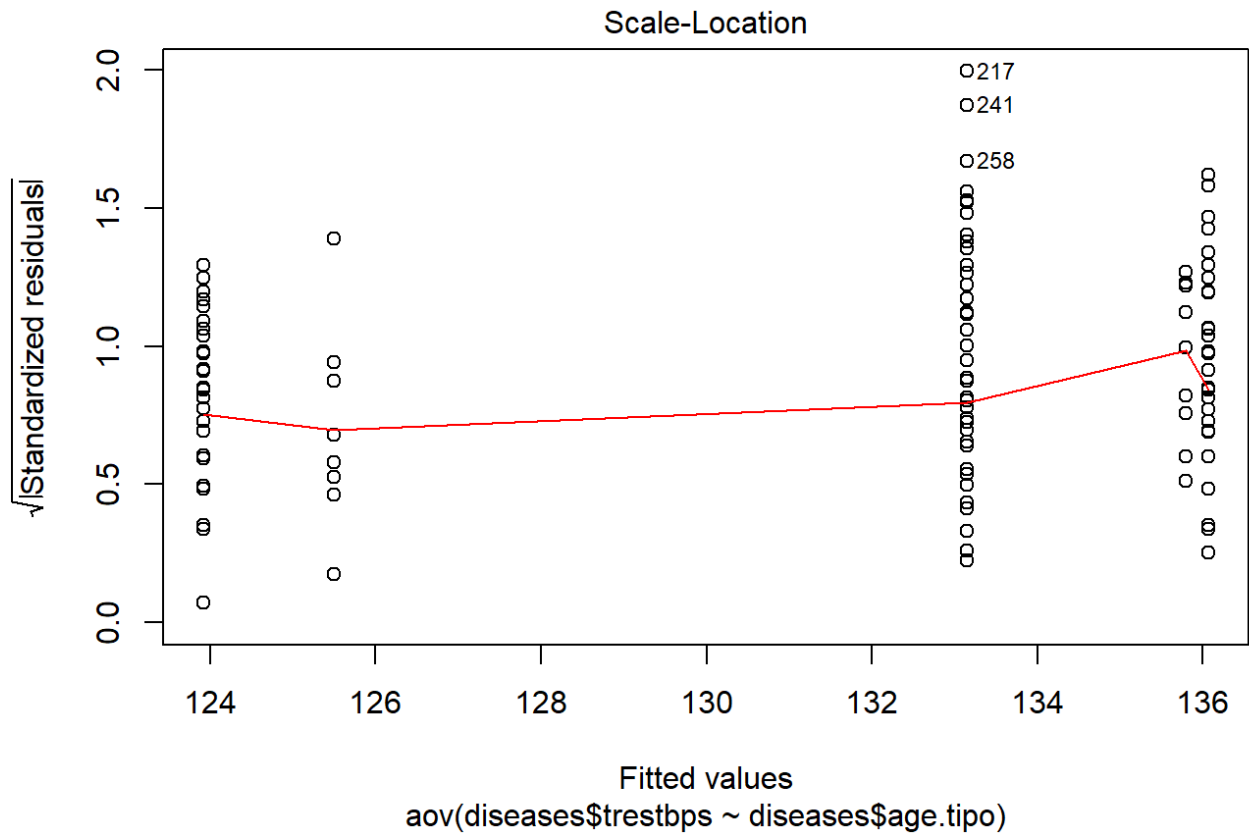
```
anova=aov(diseases$trestbps ~ diseases$age.tipo)
summary (anova)
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## diseases$age.tipo    4   6781   1695.1     5.96 0.000128 ***
## Residuals          288  81913    284.4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos que al ser p-value es inferior a 0.05 sí que hay evidencias suficientes para considerar que al menos dos medias son distintas. No obstante vamos a probar si se cumplen las condiciones para aplicar ANOVA. Revisamos los siguientes gráficos.

```
plot( anova)
```





En el gráfico 1, la representación gráfica de los residuos nos muestra falta de homocedasticidad y en el gráfico 2 qqplot vemos que los residuos no se distribuyen cercanos a la línea de la normal. Decidimos aplicar un estudio no paramétrico.

```
diseases$age.tipo <- as.factor(diseases$age.tipo)
kruskal.test(trestbps ~ age.tipo, data = diseases)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  trestbps by age.tipo
## Kruskal-Wallis chi-squared = 21.946, df = 4, p-value = 0.0002055
```

```
pairwise.wilcox.test(x = diseases$trestbps, g = diseases$age.tipo, p.adjust.method
= "holm" )
```

```
## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  diseases$trestbps and diseases$age.tipo
##
##               Mayor a 40 y menor a 50 Mayor a 50 y menor a 60
## Mayor a 50 y menor a 60 0.0081          -
## Mayor a 60 y menor a 70 0.0002          0.8864
## Mayores a 70           0.4207          1.0000
## Menores a 40           1.0000          0.7026
##
##               Mayor a 60 y menor a 70 Mayores a 70
## Mayor a 50 y menor a 60 -              -
## Mayor a 60 y menor a 70 -              -
## Mayores a 70           1.0000          -
## Menores a 40           0.1289          0.8864
##
## P value adjustment method: holm
```

También este estudio (test de Kruskal) nos dice que existe diferencias significativas, pero no entre todos los grupos de edad como nos indica el segundo test pairwise.wilcox.

Repetimos este estudio para otras variables:

```
kruskal.test(thalach ~ age.tipo, data = diseases)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  thalach by age.tipo
## Kruskal-Wallis chi-squared = 48.683, df = 4, p-value = 6.8e-10
```



```
pairwise.wilcox.test(x = diseases$thalach, g = diseases$age.tipo, p.adjust.method =
"holm" )
```

```
## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test
##
## data: diseases$thalach and diseases$age.tipo
##
##               Mayor a 40 y menor a 50 Mayor a 50 y menor a 60
## Mayor a 50 y menor a 60 0.00521 -
## Mayor a 60 y menor a 70 4.0e-06 0.02677
## Mayores a 70 0.00196 0.03960
## Menores a 40 0.03960 0.00073
##
##               Mayor a 60 y menor a 70 Mayores a 70
## Mayor a 50 y menor a 60 - -
## Mayor a 60 y menor a 70 - -
## Mayores a 70 0.10300 -
## Menores a 40 3.2e-05 0.00189
##
## P value adjustment method: holm
```

```
kruskal.test(oldpeak ~ age.tipo, data = diseases)
```

```
##
## Kruskal-Wallis rank sum test
##
## data: oldpeak by age.tipo
## Kruskal-Wallis chi-squared = 23.463, df = 4, p-value = 0.0001023
```

```
pairwise.wilcox.test(x = diseases$oldpeak, g = diseases$age.tipo, p.adjust.method =
"holm" )
```

```
## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties
```

```
##
## Pairwise comparisons using Wilcoxon rank sum test
##
## data: diseases$oldpeak and diseases$age.tipo
##
##               Mayor a 40 y menor a 50 Mayor a 50 y menor a 60
## Mayor a 50 y menor a 60 0.056 -
## Mayor a 60 y menor a 70 8.2e-05 0.068
## Mayores a 70 0.787 1.000
## Menores a 40 1.000 0.787
##               Mayor a 60 y menor a 70 Mayores a 70
## Mayor a 50 y menor a 60 - -
## Mayor a 60 y menor a 70 - -
## Mayores a 70 1.000 -
## Menores a 40 0.083 0.947
##
## P value adjustment method: holm
```

En el caso de thalach (maximo ritmo cardiaco alcanzado) vemos que practicamente entre todos los grupos de edad las medias son significativamente distintas. El resultado para oldpeak (depresión del ST inducida por el ejercicio en relación con el descanso) es más parecido a la variable trestbps (hay diferencia entre los diferentes grupos pero no entre todos entre ellos).

Las conclusiones eran previsibles, los diferentes grupos de edad tienen medias de trestbps, thalach y oldpeak significativamente distintas (entre algunos grupos entre ellos). Sobre todo en thalach vemos medias diferentes practicamente entre todos los grupos. De momento no profundizamos más en este estudio y lo retomaremos si por los modelos que realicemos nos interesa profundizar en el análisis.

2 Análisis y pruebas estadísticas

2.1 Análisis de los datos.

2.1.1 Selección de los grupos de datos que se quieren analizar/comparar (planificación de los análisis a aplicar).

La selección de datos finalmente quedará en los siguiente:

```
str(diseases)
```

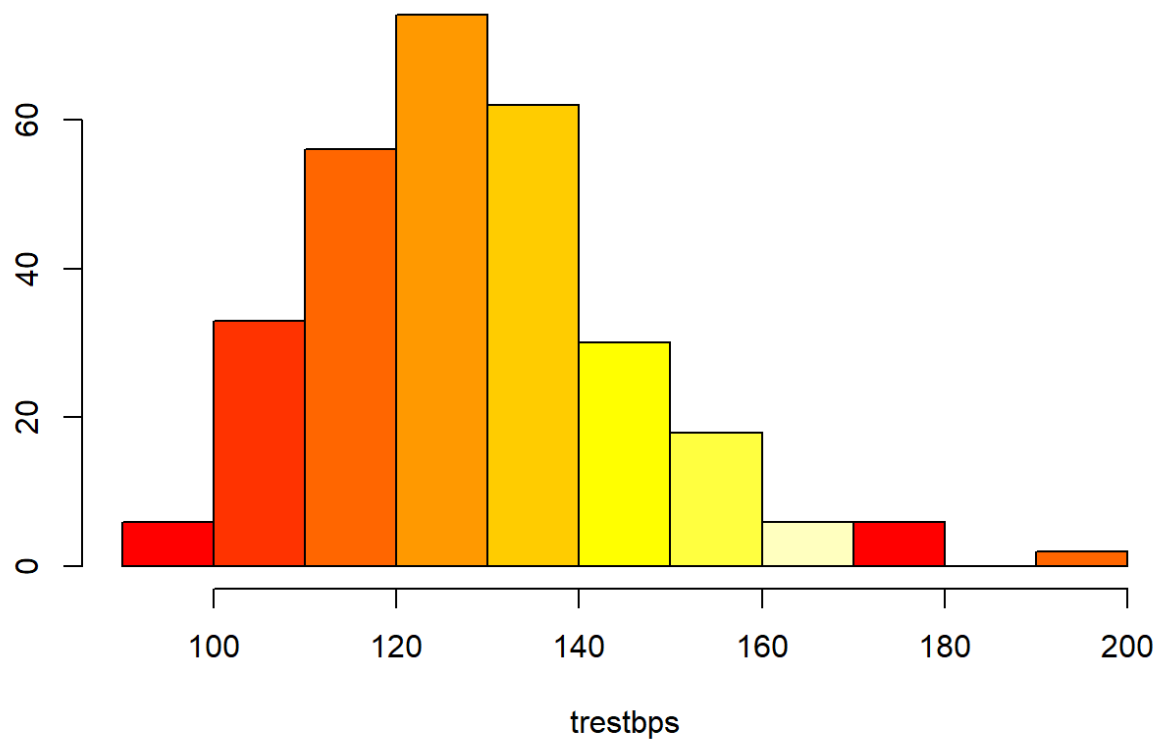
```
## 'data.frame':    293 obs. of  26 variables:
## $ age          : num  63 37 41 56 57 57 56 44 52 57 ...
## $ sex          : int  1 1 0 1 0 1 0 1 1 1 ...
## $ cp          : int  3 2 1 1 0 0 1 1 2 2 ...
## $ trestbps     : int  145 130 130 120 120 140 140 120 172 150 ...
## $ chol         : int  233 250 204 236 354 192 294 263 199 168 ...
## $ fbs         : int  1 0 0 0 0 0 0 0 1 0 ...
## $ restecg     : int  0 1 0 1 1 1 0 1 1 1 ...
## $ thalach      : int  150 187 172 178 163 148 153 173 162 174 ...
## $ exang        : int  0 0 0 0 1 0 0 0 0 0 ...
## $ oldpeak      : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
## $ slope        : int  0 0 2 2 2 1 1 2 2 2 ...
## $ ca          : int  0 0 0 0 0 0 0 0 0 0 ...
## $ target       : int  1 1 1 1 1 1 1 1 1 1 ...
## $ target.tipo  : chr   "Con Problemas Cardiacos" "Con Problemas Cardiacos" "Con
Problemas Cardiacos" "Con Problemas Cardiacos" ...
## $ age.tipo     : Factor w/ 5 levels "Mayor a 40 y menor a 50",...: 3 5 1 2 2 2 2
1 2 2 ...
## $ sex.tipo     : chr   "Masculino" "Masculino" "Femenino" "Masculino" ...
## $ cp.tipo      : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
## $ trestbps.tipo: chr   "Mayor a 140 y menor a 180" "Mayor a 100 y menor a 150"
"Mayor a 100 y menor a 150" "Mayor a 100 y menor a 150" ...
## $ chol.tipo    : chr   "Mayor a 200 y menor a 300" "Mayor a 200 y menor a 300"
"Mayor a 200 y menor a 300" "Mayor a 200 y menor a 300" ...
## $ fbs.tipo     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
## $ restecg.tipo : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
## $ thalach.tipo : chr   "Mayor a 150 y menor a 200" "Mayor a 150 y menor a 200"
"Mayor a 150 y menor a 200" "Mayor a 150 y menor a 200" ...
## $ exang.tipo   : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
## $ oldpeak.tipo : chr   "Mayor a 2 y menor a 3" "Mayor a 3 y menor a 4" "Mayor a
1 y menor a 2" "Menor a 1" ...
## $ slope.tipo   : Factor w/ 3 levels "0","1","2": 1 1 3 3 3 2 2 3 3 3 ...
## $ ca.tipo      : Factor w/ 5 levels "0","1","2","3",...: 1 1 1 1 1 1 1 1 1 1 ...
```

2.1.2 Comprobación de la normalidad y homogeneidad de la varianza.

Vamos a probar la normalidad de los datos mediante el test Shapiro–Wilk. Seleccionamos las variables a las que aplica comprobar la normalidad, que son las variables cuantitativas. Visualizamos unos gráficos en columna, que nos ayudará a intuir su distribución.

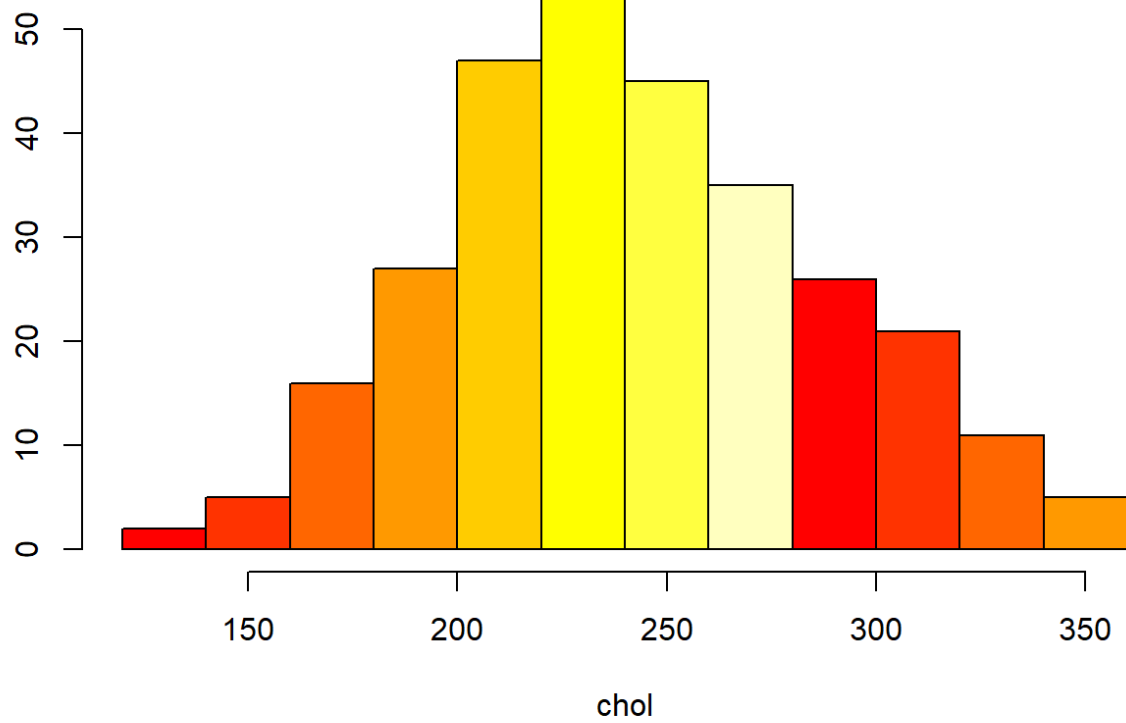
```
trestbps<-diseases$trestbps
chol<-diseases$chol
thalach<-diseases$thalach
oldpeak<-diseases$oldpeak
hist(trestbps,main = "Histograma variable trestbps",ylab=names("trestbps"),col=hea
t.colors(8))
```

Histograma variable trestbps



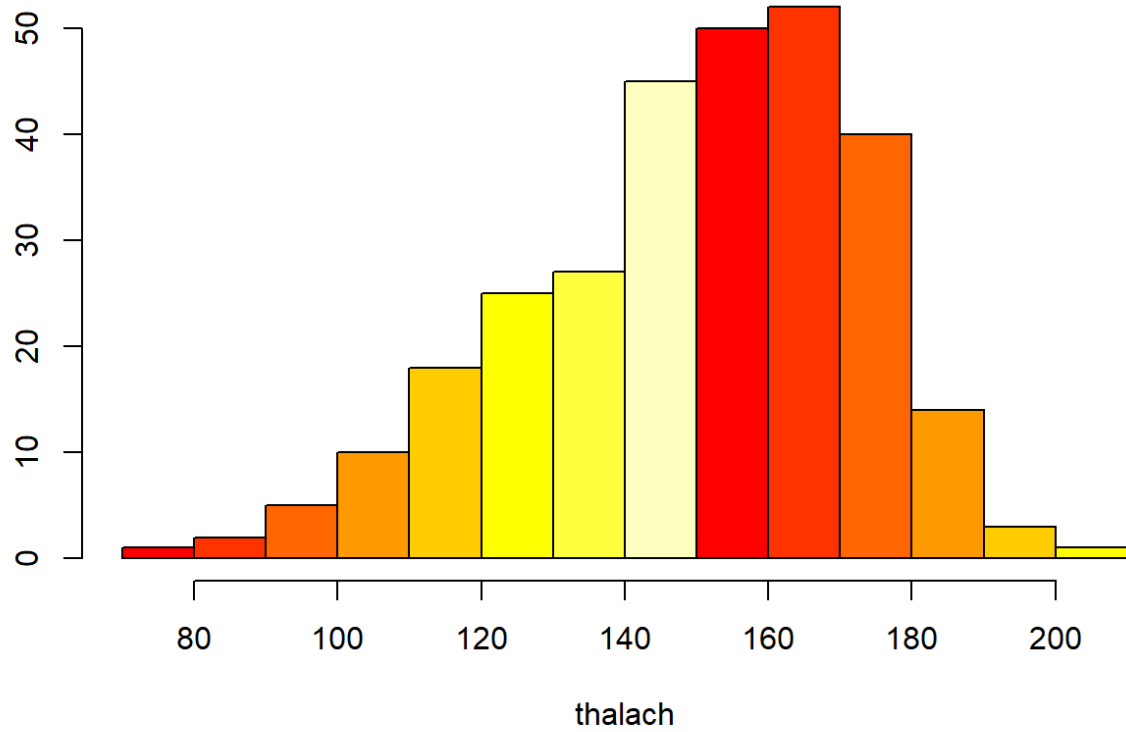
```
hist(chol,main = "Histograma variable chol",ylab=names("chol"),col=heat.colors(8))
```

Histograma variable chol



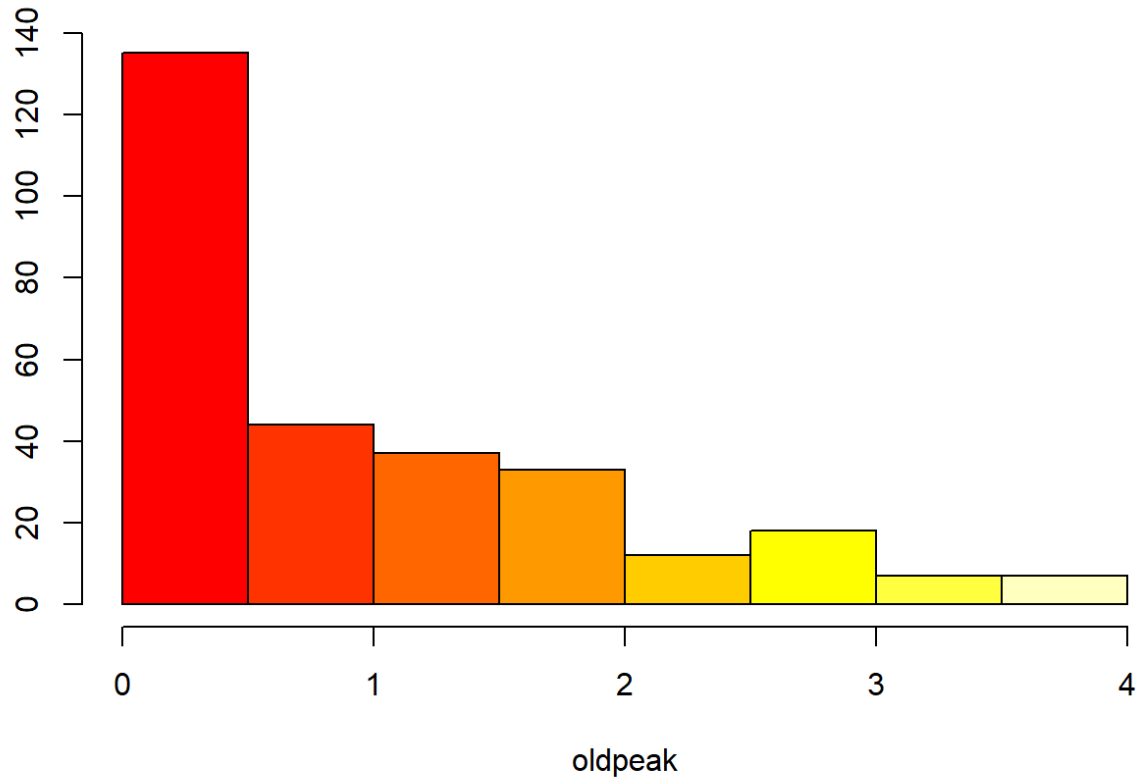
```
hist(thalach,main = "Histograma variable thalach",ylab=names("thalach"),col=heat.colors(8))
```

Histograma variable thalach



```
hist(oldpeak,main = "Histograma variable oldpeak",ylab=names("oldpeak"),col=heat.colors(8))
```

Histograma variable oldpeak



```
#trestbps<-presión arterial en reposo (en mm Hg al ingreso en el hospital)
shapiro.test(trestbps)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  trestbps
## W = 0.965, p-value = 1.571e-06
```

```
#chol<-suero colestoral en mg/dl
shapiro.test(chol)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  chol
## W = 0.99284, p-value = 0.1734
```

```
#thalach<-maximo ritmo cardiaco alcanzado
shapiro.test(thalach)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: thalach  
## W = 0.97526, p-value = 5.946e-05
```

```
#oldpeak<-depresión del ST inducida por el ejercicio en relación con el descanso.  
shapiro.test(oldpeak)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: oldpeak  
## W = 0.85349, p-value = 5.343e-16
```

Respecto a todas las variables, exceptuando la variable chol, los valores del pvalue son muy pequeños y podemos determinar que no tienen una distribución normal (se rechaza la hipótesis nula y por lo tanto asumimos que no hay normalidad en los datos). La variable chol sí podemos considerar que tiene distribución normal ($pvalue=0.1734 > 0.05$). Si vemos el gráfico de esa variable podemos ver esa característica.

A continuación trataremos el análisis de la varianza. Vamos a realizar un análisis unifactorial, donde la variable dependiente va a ser chol y la variable independiente será el sexo. En otras palabras, determinaremos si el nivel de chol (suero colestoral en mg/d) viene determinado por el sexo. Al tener muestras grandes de más de 30 no es necesario que confirmemos que la distribución de cada grupo (hombres/mujeres) es normal.

```
#tapply(diseases$chol, diseases$sex, mean)  
#variable chol<-suero colestoral en mg/dl  
fligner.test(chol ~ sex, data = diseases)
```

```
##  
## Fligner-Killeen test of homogeneity of variances  
##  
## data: chol by sex  
## Fligner-Killeen:med chi-squared = 2.9102, df = 1, p-value =  
## 0.08802
```

```
#Otra forma de realizar la prueba varianceTest(diseases$chol[diseases$sex=='0'], di  
seases$chol[diseases$sex=='1'], method = "fligner")
```

Al ser p-value mayor al nivel de significación 0.05 podemos concluir que hay diferencias significativas entre las varianzas.

Ahora repetimos la prueba para las variables trestbps, thalach y oldpeak y siendo la variable dependiente también el sexo.


```
#variable trestbps<-presión arterial en reposo (en mm Hg al ingreso en el hospital)
fligner.test(trestbps ~ sex, data = diseases)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  trestbps by sex
## Fligner-Killeen:med chi-squared = 1.1117, df = 1, p-value = 0.2917
```

```
#variable thalach<-maximo ritmo cardiaco alcanzado
fligner.test(thalach ~ sex, data = diseases)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  thalach by sex
## Fligner-Killeen:med chi-squared = 3.2289, df = 1, p-value =
## 0.07235
```

```
#variable oldpeak<-depresión del ST inducida por el ejercicio en relación con el de
scanso.
fligner.test(oldpeak ~ sex, data = diseases)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  oldpeak by sex
## Fligner-Killeen:med chi-squared = 6.6638, df = 1, p-value =
## 0.009839
```

Solo para oldpeak obtenemos un resultado diferente, en el que podemos concluir que no hay diferencias significativas entre las varianzas. Acabamos nuestro estudio de análisis de varianza aquí. Podría ampliarse con el estudio de otras variables o incluso realizando análisis multifactorial si se requiriera profundizar en algun aspecto concreto del estudio.

2.1.3 Aplicación de pruebas estadísticas para comparar los grupos de datos. En función de los datos y el objetivo del estudio, aplicar pruebas de contraste de hipótesis, correlaciones, regresiones, etc. Aplicar al menos tres métodos de análisis diferentes.

A continuación se presentarán tres modelos para el análisis de datos, Árbol de Clasificación, Regresión Logística y Naive Bayes.

Árbol de Clasificación

Vamos a aplicar un árbol de de decisión. Para ello pasamos a factor nuestra variable target. Utilizaremos las 12 primeras variables (las no transformada). Separaremos el dataset en datos de training y datos de testing. Una vez creado el modelo veremos su precisión.

```
#Pasamos a factor para crear el arbol
diseases$target.tipo = as.factor(diseases$target.tipo)
y <- diseases[,14] #target tipo
X <- diseases[,1:12]
indexes = sample(1:nrow(diseases), size=floor((2/3)*nrow(diseases)))
trainX<-X[indexes,]
trainy<-y[indexes]
testX<-X[-indexes,]
testy<-y[-indexes]
#Primero visualizamos las reglas
modelClasif <- C50::C5.0(trainX, trainy,rules=TRUE )
summary(modelClasif)
```

```
##
## Call:
## C5.0.default(x = trainX, y = trainy, rules = TRUE)
##
##
## C5.0 [Release 2.07 GPL Edition]      Mon Jun 10 20:04:57 2019
## -----
##
## Class specified by attribute `outcome'
##
## Read 195 cases (13 attributes) from undefined.data
##
## Rules:
##
## Rule 1: (65/7, lift 1.5)
##   cp > 0
##   slope > 1
##   ->  class Con Problemas Cardiacos  [0.881]
##
## Rule 2: (43/5, lift 1.5)
##   cp > 1
##   exang <= 0
##   oldpeak <= 0.9
##   ->  class Con Problemas Cardiacos  [0.867]
##
## Rule 3: (65/8, lift 1.5)
##   slope > 1
##   ca <= 0
##   ->  class Con Problemas Cardiacos  [0.866]
##
## Rule 4: (90/12, lift 1.5)
##   exang <= 0
##   ca <= 0
##   ->  class Con Problemas Cardiacos  [0.859]
##
## Rule 5: (14/2, lift 1.4)
##   cp > 0
##   trestbps <= 115
##   ->  class Con Problemas Cardiacos  [0.813]
##
## Rule 6: (39/2, lift 2.2)
##   cp <= 0
##   exang > 0
##   slope <= 1
##   ->  class Sin Problemas Cardiacos  [0.927]
##
## Rule 7: (38/2, lift 2.1)
##   trestbps > 115
##   exang > 0
```

```

## slope <= 1
## -> class Sin Problemas Cardiacos [0.925]
##
## Rule 8: (37/2, lift 2.1)
## cp <= 1
## restecg <= 0
## ca > 0
## -> class Sin Problemas Cardiacos [0.923]
##
## Rule 9: (75/22, lift 1.6)
## ca > 0
## -> class Sin Problemas Cardiacos [0.701]
##
## Default class: Con Problemas Cardiacos
##
##
## Evaluation on training data (195 cases):
##
##           Rules
##   -----
##      No      Errors
##
##      9    23(11.8%)  <<
##
##
##      (a)  (b)    <-classified as
##   ----  ----
##     103     8    (a): class Con Problemas Cardiacos
##      15    69    (b): class Sin Problemas Cardiacos
##
##
## Attribute usage:
##
## 88.21% ca
## 74.87% exang
## 71.28% cp
## 65.64% slope
## 26.67% trestbps
## 22.05% oldpeak
## 18.97% restecg
##
##
## Time: 0.0 secs

```

```

modelClasif <- C50::C5.0(trainX, trainy)
plot(modelClasif, gp = gpar(fontsize = 4))

```


Por otra parte podemos destacar algunas reglas importante que se visualizan como son:

-el 92% de los individuos que cumplen $trestbps \leq 146$ (presión arterial en reposo), $oldpeak \leq 0.7$ (depresión del ST inducida por el ejercicio en relación con el descanso) y $ca \leq 0$ (número de vasos principales coloreados por fluoroscopia) tienen problemas cardíacos.

-el 89% de los individuos con $cp \leq 0$ (con dolor en el pecho), $trestbps \leq 115$ (se refiere a presión arterial) y $ca \leq 0$ (número de vasos principales coloreados por fluoroscopia) tienen problema cardíacos.

Modelo de regresión logística

```
#Preparamos los datos en un data set de entrenamiento y otro de test
set.seed(99)
dataRL <- diseases[,1:13]
train.index <- sample(1:nrow(dataRL), size=floor((2/3)*nrow(dataRL)))
train.data <- dataRL[train.index,]
test.data <- dataRL[-train.index,]

# Creamos el modelo de regresión logística basado en target, la cual es nuestra variable a predecir
modeloRL <- glm(target~.,data =train.data,family =binomial)
summary(modeloRL)
```

```
##
## Call:
## glm(formula = target ~ ., family = binomial, data = train.data)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.4320  -0.3886   0.1733   0.6125   2.1909
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.862049   3.112209   1.562 0.118229
## age          0.008092   0.028256   0.286 0.774579
## sex         -2.330506   0.590774  -3.945 7.99e-05 ***
## cp           0.858964   0.241446   3.558 0.000374 ***
## trestbps    -0.026528   0.013759  -1.928 0.053842 .
## chol        -0.018379   0.006375  -2.883 0.003941 **
## fbs         -0.236495   0.701134  -0.337 0.735889
## restecg      0.142187   0.451666   0.315 0.752909
## thalach      0.032014   0.013587   2.356 0.018458 *
## exang        -1.495371   0.526089  -2.842 0.004477 **
## oldpeak     -0.698412   0.292205  -2.390 0.016842 *
## slope        0.298538   0.477539   0.625 0.531867
## ca          -0.656412   0.229686  -2.858 0.004265 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 266.00  on 194  degrees of freedom
## Residual deviance: 135.45  on 182  degrees of freedom
## AIC: 161.45
##
## Number of Fisher Scoring iterations: 6
```

```
# Agregamos a cada columna de nuestros datos de test la probabilidad y predicción
test.data[, "PROB_SUCCESS"] <- predict(modeloRL, newdata = test.data, type="response")
test.data[, "PREDICCIÓN"] <- ifelse(test.data[, "PROB_SUCCESS"]>=0.5, 1, 0) # si la
s probabilidades supera el 50% indicamos 1, sino 0 (predicción a la variable targe
t)
head(test.data) # mostramos una muestra de la predicción
```

```
##      age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca
## 5    57  0  0      120  354  0          1     163     1     0.6     2  0
## 7    56  0  1      140  294  0          0     153     0     1.3     1  0
## 8    44  1  1      120  263  0          1     173     0     0.0     2  0
## 9    52  1  2      172  199  1          1     162     0     0.5     2  0
## 14   64  1  3      110  211  0          0     144     1     1.8     1  0
## 17   58  0  2      120  340  0          1     172     0     0.0     2  0
##      target PROB_SUCCESS PREDICCIÓN
## 5          1    0.4199119          0
## 7          1    0.7934009          1
## 8          1    0.8815506          1
## 9          1    0.8569101          1
## 14         1    0.7283605          1
## 17         1    0.9794220          1
```

```
# Nos indica la precisión del modelo
print(sprintf("La precisión es: %.4f %%", 100*mean(test.data$PREDICCIÓN==test.data$target)))
```

```
## [1] "La precisión es: 80.6122 %"
```

```
# Mostramos la matriz de confusión con los datos obtenidos en el test
table(test.data[, "target"], test.data[, "PREDICCIÓN"], dnn=c("Actual", "Predicho"))
```

```
##      Predicho
## Actual  0  1
##      0 37 12
##      1  7 42
```

```
# Exportamos los datos de la predicción
write.csv(test.data, file="output/RegresionLogistica.csv")
```

Algoritmo Naive Bayes

```
# Instalamos la librería necesaria
if(!require(e1071)){
  install.packages('e1071', repos='http://cran.us.r-project.org')
  library(e1071)
}
```

```
## Loading required package: e1071
```



```
##  
## Attaching package: 'e1071'
```

```
## The following objects are masked from 'package:PerformanceAnalytics':  
##  
##      kurtosis, skewness
```

```
## The following objects are masked from 'package:timeDate':  
##  
##      kurtosis, skewness
```

```
# Copiamos el data set para utilizarlo en el algoritmo  
dataNV <- diseases[,c("age.tipo", "cp.tipo", "trestbps.tipo", "chol.tipo", "restecg.tipo",  
  "exang.tipo", "oldpeak.tipo", "slope.tipo", "ca.tipo", "target")]  
dataNV$target <- as.factor(dataNV$target)  
  
# Creamos nuestros datos para el modelo y test  
trainNV.index <- sample(1:nrow(dataNV), size=floor((2/3)*nrow(dataNV)))  
trainNV.data <- dataNV[trainNV.index,]  
testNV.data <- dataNV[-trainNV.index,]  
  
# Ejecutamos el algoritmo  
mod <- naiveBayes(target~ ., data = trainNV.data)  
summary(mod)
```

```
##           Length Class  Mode  
## apriori      2      table numeric  
## tables       9    -none- list  
## levels       2    -none- character  
## isnumeric    9    -none- logical  
## call         4    -none- call
```

```
# Guardamos los valores predichos en el data set de test para luego compararlos  
testNV.data[, "PREDICCION"] <- predict(mod, testNV.data)  
  
# Vemos una muestra de las predicciones  
head(testNV.data)
```

```
##          age.tipo cp.tipo          trestbps.tipo
## 2          Menores a 40          2 Mayor a 100 y menor a 150
## 3 Mayor a 40 y menor a 50          1 Mayor a 100 y menor a 150
## 6 Mayor a 50 y menor a 60          0 Mayor a 140 y menor a 180
## 10 Mayor a 50 y menor a 60          2 Mayor a 140 y menor a 180
## 11 Mayor a 50 y menor a 60          0 Mayor a 140 y menor a 180
## 16 Mayor a 50 y menor a 60          2 Mayor a 100 y menor a 150
##          chol.tipo restecg.tipo exang.tipo          oldpeak.tipo
## 2 Mayor a 200 y menor a 300          1          0 Mayor a 3 y menor a 4
## 3 Mayor a 200 y menor a 300          0          0 Mayor a 1 y menor a 2
## 6          Menor a 200          1          0          Menor a 1
## 10          Menor a 200          1          0 Mayor a 1 y menor a 2
## 11 Mayor a 200 y menor a 300          1          0 Mayor a 1 y menor a 2
## 16 Mayor a 200 y menor a 300          1          0 Mayor a 1 y menor a 2
## slope.tipo ca.tipo target PREDICCIÓN
## 2          0          0          1          0
## 3          2          0          1          1
## 6          1          0          1          1
## 10         2          0          1          1
## 11         2          0          1          1
## 16         1          0          1          1
```

Nos indica la precisión del modelo

```
print(sprintf("La precisión es: %.4f %%", 100*mean(testNV.data$PREDICCIÓN==testNV.data$target)))
```

```
## [1] "La precisión es: 79.5918 %"
```

Vemos una tabla de confusión y comparamos los datos predecidos con los reales

```
table(testNV.data$target, testNV.data$PREDICCIÓN, dnn = c("Actual", "Predicha"))
```

```
##          Predicha
## Actual  0  1
##          0 35 10
##          1 10 43
```

Exportamos los datos de la predicción

```
write.csv(testNV.data, file="output/NaiveBayes.csv")
```

Cross Validation para los modelos propuestos

Pasaremos a revisar con K iteraciones la validez de las predicciones de los modelos seleccionados. La cantidad de iteraciones elegidas es 10.

```
library(caret)
```

```
## Loading required package: lattice
```

```
folds <- createFolds(dataRL$target, k = 10)
```

```
# Cross Validation para Árbol de Clasificación
```

```
# Iteramos el método de Árbol de Clasificación en K veces para probar nuestro modelo y predecir su verdadera efectividad
```

```
cvC50ArbolClasificacion <- lapply(folds, function(x){
```

```
  trainX<-X[-x,]
```

```
  trainy<-y[-x]
```

```
  testX<-X[x,]
```

```
  testy<-y[x]
```

```
  modelClasif <- C50::C5.0(trainX, trainy)
```

```
  predicted_model <- predict( modelClasif, testX, type="class" )
```

```
  precision <- sum(predicted_model == testy) / length(predicted_model)
```

```
  return(precision)
```

```
})
```

```
#Calculamos la media aritmética de k iteraciones
```

```
precisionC50ArbolClasificacion <- mean(as.numeric(cvC50ArbolClasificacion))
```

```
print(sprintf("La precisión para Árbol de clasificación en Cross Validation: %.4f %  
%",100*precisionC50ArbolClasificacion))
```

```
## [1] "La precisión para Árbol de clasificación en Cross Validation: 75.5057 %"
```

```
# Cross Validation para Regresion Logistica
```

```
# Iteramos el método Regresion Logistica en K veces para probar nuestro modelo y predecir su verdadera efectividad
```

```
cvRegresionLogistica <- lapply(folds, function(x){
```

```
  training_fold <- dataRL[-x, ]
```

```
  test_fold <- dataRL[x, ]
```

```
  clasificador <- glm(target ~ ., family = binomial, data = training_fold)
```

```
  y_pred <- predict(clasificador, type = 'response', newdata = test_fold)
```

```
  y_pred <- ifelse(y_pred > 0.5, 1, 0)
```

```
  #y_pred <- factor(y_pred, levels = c("0", "1"), labels = c("NoPulsar", "Pulsar"))
```

```
  cm <- table(test_fold$target, y_pred)
```

```
  precision <- (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] +cm[1,2] + cm[2,1])
```

```
  return(precision)
```

```
})
```

```
#Calculamos la media aritmética de k iteraciones
```

```
precisionRegresionLogistica <- mean(as.numeric(cvRegresionLogistica))
```

```
print(sprintf("La precisión de Regresión Logística en Cross Validation: %.4f %%",100*precisionRegresionLogistica))
```

```
## [1] "La precisión de Regresión Logística en Cross Validation: 81.2414 %"
```

```
# Cross Validation para NaiveBayes
# Iteramos el método de NaiveBayes en K veces para probar nuestro modelo y predecir
  su verdadera efectividad
cvNaiveBayes <- lapply(folds, function(x){
  training_fold <- dataNV[-x, ]
  test_fold <- dataNV[x, ]
  clasificador <- naiveBayes(target ~ ., data = training_fold)
  y_pred <- predict(clasificador, newdata = test_fold)
  cm <- table(test_fold$target, y_pred)
  precision <- (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
  return(precision)
})

#Calculamos la media aritmética de k iteraciones
precisionNaiveBayes <- mean(as.numeric(cvNaiveBayes))
print(sprintf("La precisión para NaiveBayes en Cross Validation: %.4f %%",100*precisionNaiveBayes))
```

```
## [1] "La precisión para NaiveBayes en Cross Validation: 79.8736 %"
```

A continuación se mostrarán los métodos ordenados descendientemente por el porcentaje y error de predicción, la varianza y desviación estandar de todos los modelos que se sometieron al cross validation.

```
vectorMedia = c(precisionC50ArbolClasificacion*100,precisionRegresionLogistica*100,
precisionNaiveBayes*100)
vectorError = c(100-precisionC50ArbolClasificacion*100,100-precisionRegresionLogistica*100,100-precisionNaiveBayes*100)
vectorVarianza = c(var(as.numeric(cvC50ArbolClasificacion)*100), var(as.numeric(cvRegresionLogistica)*100), var(as.numeric(cvNaiveBayes)*100))
vectorDesviacion = c(sd(as.numeric(cvC50ArbolClasificacion)*100), sd(as.numeric(cvRegresionLogistica)*100), sd(as.numeric(cvNaiveBayes)*100))
M1 = cbind(c("Árbol Clasificacion", "Regresion Logistica","Naive Bayes"),vectorMedia,vectorError, vectorVarianza,vectorDesviacion)
colnames(M1) <- c("Modelo", "Porcentaje Predicción", "Error", "Varianza", "Desviación")
```

```
# Ordenamos y mostramos los resultados
M1[order(M1[, "Porcentaje Predicción"],decreasing = TRUE), ]
```

##	Modelo	Porcentaje Predicción	Error
## [1,]	"Regresion Logistica"	"81.2413793103448"	"18.7586206896552"
## [2,]	"Naive Bayes"	"79.8735632183908"	"20.1264367816092"
## [3,]	"Árbol Clasificacion"	"75.5057471264368"	"24.4942528735632"
##	Varianza	Desviación	
## [1,]	"25.7095462485871"	"5.07045818921595"	
## [2,]	"39.8187049514834"	"6.31020641116306"	
## [3,]	"75.3143670821039"	"8.67838505034802"	

Los resultados del Cross Validation nos puede dar la visión por cuál de los modelos nos podemos inclinar, entregando un mejor resultado en la verisidad de la predicción. Para selección del modelo apropiado, podemos tomar como referencia el porcentaje de predicción +- la desviación estandar.

Probando la eliminación slope por correlación entre oldspeak

Ahora veremos eliminado la variable slope, probando si la correlación entre oldspeak es suficiente como para eliminarla por redundancia.

```
X$slope <- NULL

# Cross Validation para Árbol de Clasificación sin variable slope
cvC50ArbolClasificacion <- lapply(folds, function(x){
  trainX<-X[-x,]
  trainy<-y[-x]
  testX<-X[x,]
  testy<-y[x]
  modelClasif <- C50::C5.0(trainX, trainy)
  predicted_model <- predict( modelClasif, testX, type="class" )
  precision <- sum(predicted_model == testy) / length(predicted_model)
  return(precision)
})
#Calculamos la media aritmética de k iteraciones
precisionC50ArbolClasificacion <- mean(as.numeric(cvC50ArbolClasificacion))
print(sprintf("La precisión para Árbol de clasificación en Cross Validation: %.4f %",100*precisionC50ArbolClasificacion))
```

```
## [1] "La precisión para Árbol de clasificación en Cross Validation: 76.8966 %"
```

```

dataRL <- diseases[,1:13]
dataRL$slope <- NULL
# Cross Validation para Regresion Logistica sin variable slope
cvRegresionLogistica <- lapply(folds, function(x){
  training_fold <- dataRL[-x, ]
  test_fold <- dataRL[x, ]
  clasificador <- glm(target ~ ., family = binomial, data = training_fold)
  y_pred <- predict(clasificador, type = 'response', newdata = test_fold)
  y_pred <- ifelse(y_pred > 0.5, 1, 0)
  #y_pred <- factor(y_pred, levels = c("0", "1"), labels = c("NoPulsar", "Pulsar"))
  cm <- table(test_fold$target, y_pred)
  precision <- (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] +cm[1,2] + cm[2,1])
  return(precision)
})
#Calculamos la media aritmética de k iteraciones
precisionRegresionLogistica <- mean(as.numeric(cvRegresionLogistica))
print(sprintf("La precisión de Regresión Logística en Cross Validation: %.4f %%",10
0*precisionRegresionLogistica))

```

```
## [1] "La precisión de Regresión Logística en Cross Validation: 82.2759 %"
```

```

dataNV <- diseases[,c("age.tipo","cp.tipo","trestbps.tipo","chol.tipo","restecg.tipo",
"exang.tipo","oldpeak.tipo","slope.tipo","ca.tipo","target")]
dataNV$target <- as.factor(dataNV$target)
dataNV$slope.tipo<- NULL
# Cross Validation para NaiveBayes sin variable slope
cvNaiveBayes <- lapply(folds, function(x){
  training_fold <- dataNV[-x, ]
  test_fold <- dataNV[x, ]
  clasificador <- naiveBayes(target ~ ., data = training_fold)
  y_pred <- predict(clasificador, newdata = test_fold)
  cm <- table(test_fold$target, y_pred)
  precision <- (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] +cm[1,2] + cm[2,1])
  return(precision)
})

#Calculamos la media aritmética de k iteraciones
precisionNaiveBayes <- mean(as.numeric(cvNaiveBayes))
print(sprintf("La precisión para NaiveBayes en Cross Validation: %.4f %%",100*precisionNaiveBayes))

```

```
## [1] "La precisión para NaiveBayes en Cross Validation: 79.8851 %"
```

```
vectorMedia = c(precisionC50ArbolClasificacion*100,precisionRegresionLogistica*100,
precisionNaiveBayes*100)
vectorError = c(100-precisionC50ArbolClasificacion*100,100-precisionRegresionLogistica*100,100-precisionNaiveBayes*100)
vectorVarianza = c(var(as.numeric(cvC50ArbolClasificacion)*100), var(as.numeric(cvRegresionLogistica)*100), var(as.numeric(cvNaiveBayes)*100))
vectorDesviacion = c(sd(as.numeric(cvC50ArbolClasificacion)*100), sd(as.numeric(cvRegresionLogistica)*100), sd(as.numeric(cvNaiveBayes)*100))
M1 = cbind(c("Árbol Clasificacion", "Regresion Logistica","Naive Bayes"),vectorMedia,vectorError, vectorVarianza,vectorDesviacion)
colnames(M1) <- c("Modelo", "Porcentaje Predicción", "Error", "Varianza", "Desviación")
# Ordenamos y mostramos los resultados
M1[order(M1[, "Porcentaje Predicción"],decreasing = TRUE), ]
```

```
##      Modelo      Porcentaje Predicción Error
## [1,] "Regresion Logistica" "82.2758620689655" "17.7241379310345"
## [2,] "Naive Bayes"      "79.8850574712644" "20.1149425287356"
## [3,] "Árbol Clasificacion" "76.8965517241379" "23.1034482758621"
##      Varianza      Desviación
## [1,] "37.4298674417581" "6.1179953777163"
## [2,] "41.1561779774225" "6.41530809684325"
## [3,] "83.1138709061817" "9.11668091501407"
```

Como podemos observar, ha aumentado levemente la precisión en la predicción al eliminar la variable slope, aunque la desviación estandar también aumento levemente, por lo que no podemos considerar relevante la eliminación de la variable en nuestros modelos predictivos.

2.2 Representación de los resultados a partir de tablas y gráficas.

2.3 Resolución del problema. A partir de los resultados obtenidos, ¿cuáles son las conclusiones? ¿Los resultados permiten responder al problema?

Lo que se quería con este estudio era por una parte detectar características de los pacientes que infuyen en que se parezca la enfermedad cardíaca o no. Por otra parte poder predecir de una manera precisa si los individuos con unas características concretas sufren la enfermedad en un porcentaje alto.

El primer punto lo hemos conseguido responder con el análisis de los datos visuales y con la ayuda de algun análisis como la creación del árbol de clasificación que nos ha dado tambien una reglas. Por ejemplo, vemos que por diferenciando por sexo las mujeres tienen un más problemas cardíacos en porcentaje y que a menos oldpeak la probabilidad de sufrir la enfermedad es mayor. Las reglas obtenidas mediante el modelo C5.0 nos ha dado la información, como la siguiente:

-individuos que cumplen $trestbps \leq 146$ (presión arterial en reposo), $oldpeak \leq 0.7$ (depresión del ST inducida por el ejercicio en relación con el descanso) y $ca \leq 0$ (número de vasos principales coloreados por fluoroscopia) tienen problemas cardíacos en un 92 %

El segundo objetivo, el de obtener un modelo de predicción también ha sido conseguido. Se ha obtenido un modelo de regresión logística con más del 80% de precisión para predecir si una persona parece la enfermedad o no.

Evidentemente, si se necesitara profundizar en algún aspecto se tendría que hacer una revisión para ajustar los modelos o buscar otros que se adecuen a las necesidades.

Para resolver las problemáticas hemos tenido que realizar las tareas habituales en un proyecto de datamining como son la limpieza y transformación de datos, la visualización y la generación de modelos.

2.4 Código: Hay que adjuntar el código, preferiblemente en R, con el que se ha realizado la limpieza, análisis y representación de los datos. Si lo preferís, también podéis trabajar en Python.

El código lo pueden encontrar en <https://github.com/fmorenono/practica2>
(<https://github.com/fmorenono/practica2>)

2.5 Referencias

Referenciamos a continuación fuentes de información utilizado durante la realización de la práctica.

-El Data Set seleccionado está disponible en: <https://www.kaggle.com/ronitf/heart-disease-uci>
(<https://www.kaggle.com/ronitf/heart-disease-uci>)

-Estudio ANOVA: https://rpubs.com/Joaquin_AR/219148 (https://rpubs.com/Joaquin_AR/219148)

-Test de Kruskal-Wallis: https://rpubs.com/Joaquin_AR/219504 (https://rpubs.com/Joaquin_AR/219504)

Contribuciones	Firma
Investigación previa	Francisco Moreno - Victor Vásquez
Redacción de las respuestas	Francisco Moreno - Victor Vásquez
Desarrollo código	Francisco Moreno - Victor Vásquez