

Dataset Regresión: Abalone

Ricardo Ignacio Shepstone Aramburu

Índice

Análisis exploratorio de datos	1
Definición del problema	1
Preparación de los datos	2
Descripción de los datos	2
Procesamiento de los datos	2
Resumen de los datos	7
Descomposición de atributos complicados	75
Búsqueda de datos redundantes	76
Transformación de datos	79
Conclusiones	83
Regresión	84
Obtención de modelos de regresión lineal simple	84
Obtención de modelos de regresión lineal múltiple	87
Aplicar k-nn y el mejor modelo de regresión múltiple obtenido con validación cruzada	93
Comparativa de algoritmos de regresión múltiple	95

Análisis exploratorio de datos

Definición del problema

El abulón (oreja del mar) es un molusco cuya concha es larga, plana y generalmente de forma ovalada. Para determinar su edad, normalmente se debe de cortar la concha, pulir, teñir con un colorante y examinar bajo un microscopio, para contar el número de anillos que se van formando conforme la cocha crece. Puesto que ciertos anillos son difíciles de contar, se determina que sumar 1.5 al número de anillos contados es una buena aproximación de la edad del individuo.

Este método para determinar la edad de un abulón es complejo y tedioso, por lo que es de especial interés intentar determinar la edad tomando otro tipo de medidas. En este dataset se dispondrá de ciertas medidas físicas, como dimensiones y pesos, así como el número de anillos de un conjunto de individuos; con el fin de intentar modelizar una regresión para predecir la edad (número de anillos) a partir del sexo y las medidas físicas.

En cuanto a la dependencia de las variables, tenemos que tener en cuenta que en la mayoría de las especies, un individuo crece en tamaño y aumenta en peso a lo largo de su vida, hasta llegar a cierto límite. Estas variables a su vez son dependientes entre ellas, ya que un individuo de mayor tamaño tendrá un peso mayor. Por otro lado, la edad no está directamente relacionada con el sexo, pero debido al dimorfismo sexual que existe en la mayoría de especies, el tamaño de un individuo puede verse afectado por su sexo en mayor o menor grado según la especie. En este caso particular, la variable de sexo tiene un tercer valor "Infant" que puede proporcionar cierta información sobre la edad del individuo, por lo que esta variable también tiene cierto grado de dependencia con el resto.

En base a esta información se procede a plantear ciertas cuestiones e **hipótesis**:

- El tamaño y el peso aumentan con la edad.
- Si es así, ¿Qué relación hay con respecto al número de anillos?
- ¿Alcanzan estas especies un límite de tamaño y peso?
- ¿Cómo es la relación entre las variables de dimensión y las de peso?
- ¿Aumentan todas las variables de tamaño en la misma proporción (están correlacionadas)?
- La variable sexo influye sobre las medidas físicas del individuo.
- ¿Hay dimorfismo sexual? ¿En qué grado?

Preparación de los datos

Descripción de los datos

A partir del dataset de abalone se puede construir un data frame que consta de 4177 observaciones y 9 variables. Tanto en el archivo de texto proporcionado con los datos, como en la descripción del dataset que aparece en el repositorio de UCI (<https://archive.ics.uci.edu/ml/datasets/abalone>), se puede obtener la siguiente lista con información sobre las variables:

Nombre	Tipo de dato	Unidades	Descripción
Sex	Categórico nominal		Macho (M), hembra (F) e Infante (I).
Length	Cuantitativo continuo	mm	La medida más larga de la concha.
Diameter	Cuantitativo continuo	mm	Medida perpendicular a la longitud.
Height	Cuantitativo continuo	mm	Altura con la vianda.
Whole weight	Cuantitativo continuo	gramos	Peso del abulón.
Shucked weight	Cuantitativo continuo	gramos	Peso de la vianda.
Viscera weight	Cuantitativo continuo	gramos	Peso de vísceras tras sangrado.
Shell weight	Cuantitativo continuo	gramos	Peso de la concha tras secado.
Rings	Cuantitativo discreto		Número de anillos.

Como se ha mencionado anteriormente, la variable de salida será el número de anillos del abulón (“Rings”), mientras que las variables de entrada serán el resto de variables.

Procesamiento de los datos

Importación de paquetes y del dataset.

El primer paso sería incluir los paquetes que utilizaremos y cargar el dataset con el que se va a trabajar.

```
require(tidyverse)
require(readr)
require(moments)
require(car)
require(corrplot)
require(fastDummies)
require(FactoMineR)
require(factoextra)

# Cargamos dataset
abalone.raw <- read.csv("Input/abalone/abalone.dat", comment.char="@", header=FALSE)
# Realizamos una copia con la que se trabajará
abalone <- abalone.raw
```

Realizamos las debidas comprobaciones, y obtenemos los datos de interés para el apartado de descripción de los datos.

```

#comprobamos número de columnas, filas y si se han cargado bien los datos, la estructura de estos
nrow(abalone)

## [1] 4177
ncol(abalone)

## [1] 9
str(abalone)

## 'data.frame': 4177 obs. of 9 variables:
## $ V1: int 3 2 3 1 3 2 2 2 1 1 ...
## $ V2: num 0.4 0.635 0.37 0.68 0.375 0.58 0.525 0.63 0.565 0.605 ...
## $ V3: num 0.305 0.5 0.27 0.54 0.285 0.475 0.38 0.495 0.44 0.465 ...
## $ V4: num 0.1 0.15 0.09 0.155 0.09 0.155 0.14 0.19 0.125 0.165 ...
## $ V5: num 0.342 1.376 0.185 1.534 0.254 ...
## $ V6: num 0.176 0.649 0.07 0.671 0.119 ...
## $ V7: num 0.0625 0.361 0.0425 0.379 0.0595 ...
## $ V8: num 0.0865 0.31 0.065 0.384 0.0675 ...
## $ V9: int 7 10 7 10 6 10 14 10 9 13 ...

summary(abalone)

##      V1          V2          V3          V4
## Min.   :1.000   Min.   :0.075   Min.   :0.0550   Min.   :0.0000
## 1st Qu.:1.000   1st Qu.:0.450   1st Qu.:0.3500   1st Qu.:0.1150
## Median :2.000   Median :0.545   Median :0.4250   Median :0.1400
## Mean   :1.955   Mean   :0.524   Mean   :0.4079   Mean   :0.1395
## 3rd Qu.:3.000   3rd Qu.:0.615   3rd Qu.:0.4800   3rd Qu.:0.1650
## Max.   :3.000   Max.   :0.815   Max.   :0.6500   Max.   :1.1300
##             V5          V6          V7          V8
## Min.   :0.0020   Min.   :0.0010   Min.   :0.0005   Min.   :0.0015
## 1st Qu.:0.4415   1st Qu.:0.1860   1st Qu.:0.0935   1st Qu.:0.1300
## Median :0.7995   Median :0.3360   Median :0.1710   Median :0.2340
## Mean   :0.8287   Mean   :0.3594   Mean   :0.1806   Mean   :0.2388
## 3rd Qu.:1.1530   3rd Qu.:0.5020   3rd Qu.:0.2530   3rd Qu.:0.3290
## Max.   :2.8255   Max.   :1.4880   Max.   :0.7600   Max.   :1.0050
##             V9
## Min.   : 1.000
## 1st Qu.: 8.000
## Median : 9.000
## Mean   : 9.934
## 3rd Qu.:11.000
## Max.   :29.000

```

Cambio de formato de los datos para trabajar con ellos.

Cambiamos nombres de las variables para poder trabajar con ellas, en la documentación también se explica que las variables numéricas se han dividido entre doscientos, por lo que hay que revertir este cambio para apreciar las verdaderas magnitudes.

```

colnames(abalone) <- c('Sex', 'Length', 'Diameter', 'Height', 'Whole_weight',
                       'Shucked_weight', 'Viscera_weight', 'Shell_weight', 'Rings')
abalone <- mutate_if(abalone, is.double, funs(. * 200))

```

Se cambia la variable “Sex” para incluir factores.

```
# factores en Sex
abalone <- abalone %>% mutate(Sex=factor(Sex, levels = c(1,2,3),
                                             labels = c('M', 'F', 'I')))
```

Comprobamos como han quedado los datos:

```
# comprobamos
head(abalone)
```

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight
## 1	I	80	61	20	68.3	35.2	12.5
## 2	F	127	100	30	275.2	129.9	72.2
## 3	I	74	54	18	37.1	14.0	8.5
## 4	M	136	108	31	306.8	134.2	75.8
## 5	I	75	57	18	50.9	23.8	11.9
## 6	F	116	95	31	194.8	86.1	46.0
Shell_weight Rings							
## 1		17.3	7				
## 2		62.0	10				
## 3		13.0	7				
## 4		76.8	10				
## 5		13.5	6				
## 6		57.0	10				

```
# Así es como quedan las variables:
str(abalone)
```

```
## 'data.frame': 4177 obs. of 9 variables:
## $ Sex : Factor w/ 3 levels "M","F","I": 3 2 3 1 3 2 2 2 1 1 ...
## $ Length : num 80 127 74 136 75 116 105 126 113 121 ...
## $ Diameter : num 61 100 54 108 57 95 76 99 88 93 ...
## $ Height : num 20 30 18 31 18 31 28 38 25 33 ...
## $ Whole_weight : num 68.3 275.2 37.1 306.8 50.9 ...
## $ Shucked_weight: num 35.2 129.9 14 134.2 23.8 ...
## $ Viscera_weight: num 12.5 72.2 8.5 75.8 11.9 46 29.5 42.3 36.5 49.5 ...
## $ Shell_weight : num 17.3 62 13 76.8 13.5 57 42 32.5 43 68 ...
## $ Rings : int 7 10 7 10 6 10 14 10 9 13 ...
```

Limpieza de los datos: missing values

Buscamos missing values y duplicados.

```
# Comprobación de missing values
sum(is.na(abalone))
```

```
## [1] 0
sum(is.null(abalone))
```

```
## [1] 0
# comprobamos si hay duplicados
sum(duplicated(abalone))
```

```
## [1] 0
```

A la hora de ver el resumen de nuestros datos, podemos apreciar que el mínimo y el máximo de la variable “Height” toman valores extraños. Los valores de cero se tratarán como missing values, por tener valores de longitud y diámetro normales. Mientras que los valores grandes de esta variable se pueden tratar como datos

atípicos, aunque como veremos en las gráficas, este valor máximo extremo no correspondería con un outlier natural, si no más bien como un error.

```
# El min de Height muestra un valor extraño
# (estos dos pueden ser missing values)
abalone %>% filter(Height==0)

##   Sex Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## 1   I     86        68      0       85.6          41.3         17.2
## 2   I     63        46      0       26.8          11.5          5.7
##   Shell_weight Rings
## 1        23.0     8
## 2        70.1     6

# Podemos eliminar estos dos valores, ya que tienen valores de longitud y
# diámetro, pero no de altura
abalone <- abalone[-which(abalone$Height==0),]
```

Se ha optado por prescindir de estas dos observaciones, ya que se dispone de un número grande de elementos en el dataset.

Limpieza de los datos: anomalías en una dimensión

Para un problema de regresión dado, el estudio de las anomalías de un dataset es importante, por ejemplo para una regresión lineal estas anomalías pueden afectar al cálculo de la recta por utilizar medidas como la media que son muy sensibles a datos atípicos. Para este dataset se ha optado por calcular estas anomalías univariadas mediante el rango intercuartílico, y obtener sus posiciones para incluirlas o descartarlas del estudio de las variables.

Para la obtención de las anomalías, se calculan los cuartiles y el rango intercuartílico para determinar los límites superior e inferior. El valor de estos límites determina si el valor de una observación es atípico para una variable determinada.

```
# Outliers en una dimensión
# Calculamos rango intercuartílico de todas las variables
abalone.IQR <- abalone %>% select(-Sex) %>% apply(2, IQR)
abalone.Quartiles <- abalone %>% select(-Sex) %>% apply(2, quantile,c(0.25,0.75))
Upper.limit <- abalone.Quartiles[2,]+1.5*abalone.IQR
Upper.limit
```

```
##           Length        Diameter        Height Whole_weight Shucked_weight
## 1    172.500     135.000      48.000      444.075      195.125
##   Viscera_weight   Shell_weight      Rings
## 1    98.450      125.375     15.500
Lower.limit <- abalone.Quartiles[1,]-1.5*abalone.IQR
Lower.limit
```

```
##           Length        Diameter        Height Whole_weight Shucked_weight
## 1    40.500      31.000      8.000      -124.925      -57.475
##   Viscera_weight   Shell_weight      Rings
## 1   -29.150     -33.625      3.500
```

Con los límites establecidos, se calculan las posiciones de las observaciones que toman valores fuera del rango establecido para cada variable.

```
# Obtenemos los outliers para cada variable y calculamos sus posiciones
# Length
Length.outliers <- which(abalone$Length>Upper.limit['Length'] |
                           abalone$Length<Lower.limit['Length'])
```

```

# Diameter
Diameter.outliers <- which(abalone$Diameter>Upper.limit['Diameter'] |
                           abalone$Diameter<Lower.limit['Diameter']) 

# Height
Height.outliers <- which(abalone$Height>Upper.limit['Height'] |
                           abalone$Height<Lower.limit['Height'])

# Whole_weight
Whole_weight.outliers <- which(abalone$Whole_weight>Upper.limit['Whole_weight'] |
                           abalone$Whole_weight<Lower.limit['Whole_weight'])

# Shucked_weight
Shucked_weight.outliers <- which(abalone$Shucked_weight>Upper.limit['Shucked_weight'] |
                           abalone$Shucked_weight<Lower.limit['Shucked_weight'])

# Viscera_weight
Viscera_weight.outliers <- which(abalone$Viscera_weight>Upper.limit['Viscera_weight'] |
                           abalone$Viscera_weight<Lower.limit['Viscera_weight'])

# Shell_weight
Shell_weight.outliers <- which(abalone$Shell_weight>Upper.limit['Shell_weight'] |
                           abalone$Shell_weight<Lower.limit['Shell_weight'])

# Rings
Rings.outliers <- which(abalone$Rings>Upper.limit['Rings'] |
                           abalone$Rings<Lower.limit['Rings'])

```

Estos vectores de posición nos permitirán descartar los outliers para cada variable a voluntad. También se ha calculado un vector que incluya las posiciones de todas las observaciones cuyas variables tomen al menos un valor atípico, es decir, se han calculado todas las observaciones atípicas. Veremos también que cantidad de outliers tenemos para tener una idea de la proporción de datos atípicos en el dataset.

Comprobamos cantidad de outliers

```

abalone.outliers <- unique(c(Length.outliers, Diameter.outliers, Height.outliers,
                           Whole_weight.outliers, Shucked_weight.outliers,
                           Viscera_weight.outliers, Shell_weight.outliers,
                           Rings.outliers))

length(abalone.outliers)

## [1] 394

```

Irregularidades

Cabe destacar que al inspeccionar el dataset en detalle se ha encontrado lo que podría parecer una irregularidad en la variable “Sex”, y es que hay ciertas observaciones que toman el valor “infant” con un mayor número de anillos que otras observaciones “male” o “female”.

Si agrupamos por sexo ,y visualizamos el máximo y mínimo del número de anillos por grupo, podemos apreciar esta irregularidad, incluso excluyendo los outliers univariados.

valores extraños en infants

```

abalone[abalone.outliers,] %>% group_by(Sex) %>% summarise(min.Rings=min(Rings),
                                                               max.rings=max(Rings))

```

```

## # A tibble: 3 x 3
##   Sex    min.Rings max.rings
##   <fct>     <int>     <int>
## 1 M          3        27
## 2 F          8        29
## 3 I          1        21

```

Como veremos en las representaciones gráficas de las variables, estos valores no se tratan de valores puntuales, y existe cierto solapamiento de las distribuciones. Las únicas explicaciones razonables que podría formular, sin ser un experto en la materia, serían que dependiendo de la especie de abulón a la que pertenece un individuo o el medio en el que se desarrolla, tardaría más o menos en llegar a la madurez sexual para ser catalogado por sexo; o bien, que para cierto grupo de individuos se haya hecho complicado determinar su sexo y se hayan catalogado en el grupo “infant”.

Resumen de los datos

Una vez procesados los datos pasamos al estudio de estos, ya sea mediante la estadística descriptiva con diferentes medidas numéricas, o la representación gráfica, tanto de cada variable como de las posibles combinaciones entre ellas. Con “summary()” podríamos visualizar algunas de ellas para cada variable, pero se ha obtenido por su cálculo para almacenarlas en vectores por si su uso hiciese falta.

```
# Con summary() podemos obtener la media y cuartiles
summary(abalone)
```

```
##   Sex      Length      Diameter      Height      Whole_weight
## M:1528  Min.   : 15.0  Min.   :11.00  Min.   : 2.00  Min.   : 0.40
## F:1307  1st Qu.: 90.0  1st Qu.: 70.00  1st Qu.:23.00  1st Qu.:88.45
## I:1340  Median :109.0  Median : 85.00  Median :28.00  Median :160.00
##          Mean   :104.8  Mean   : 81.59  Mean   :27.92  Mean   :165.80
##          3rd Qu.:123.0  3rd Qu.: 96.00  3rd Qu.:33.00  3rd Qu.:230.70
##          Max.   :163.0  Max.   :130.00  Max.   :226.00  Max.   :565.10
##   Shucked_weight  Viscera_weight  Shell_weight      Rings
##   Min.   : 0.20  Min.   : 0.10  Min.   : 0.30  Min.   : 1.000
##   1st Qu.: 37.25 1st Qu.: 18.70  1st Qu.: 26.00  1st Qu.: 8.000
##   Median : 67.20  Median : 34.20  Median : 46.80  Median : 9.000
##   Mean   : 71.90  Mean   : 36.13  Mean   : 47.77  Mean   : 9.935
##   3rd Qu.:100.40  3rd Qu.: 50.60  3rd Qu.: 65.75  3rd Qu.:11.000
##   Max.   :297.60  Max.   :152.00  Max.   :201.00  Max.   :29.000
```

Medidas de tendencia central

Nos permiten obtener cierta idea de centro de nuestros datos.

```
# como lista:
abalone.mean.median <- abalone %>% select(-Sex) %>%
  apply(2,function(x){list(media=mean(x), mediana=median(x))})
```

Como vector

```
abalone.mean <- abalone %>% select(-Sex) %>% map_dbl(mean)
abalone.median <- abalone %>% select(-Sex) %>% map_dbl(median)
```

```
abalone.mean
```

```
##      Length      Diameter      Height      Whole_weight Shucked_weight
## 104.81293     81.58802    27.91665     165.80091      71.89528
## Viscera_weight Shell_weight      Rings
## 36.13054      47.76675    9.93509
```

```
abalone.median
```

```
##      Length      Diameter      Height      Whole_weight Shucked_weight
## 109.0          85.0          28.0        160.0          67.2
## Viscera_weight Shell_weight      Rings
## 34.2           46.8          9.0
```

Ya con la media y la mediana, obtenemos cierta idea sobre la asimetría en las distribuciones de nuestras variables. Puesto que las medidas de dimensiones tienen una media menor a la mediana, el “skewness” es negativo. Mientras que en el resto de las variables ocurre lo contrario y el “skewness” es positivo.

Medidas de dispersión

Estas medidas nos proporcionan información de la dispersión de nuestros datos, y de la distancia a la que se encuentran del centro. Sus cálculos son sencillos, contando que además los cálculos para los cuartiles y el rango intercuartílico se ha hecho anteriormente para el estudio de los datos atípicos.

```
# Medidas de dispersión
# Para el Rango, máximo y mínimo:
abalone.min.max <- abalone %>% select(-Sex) %>% apply(2,range)
abalone.range <- abalone.min.max[2,]-abalone.min.max[1,]
abalone.min.max.range <- rbind(abalone.min.max, abalone.range)
rownames(abalone.min.max.range) <- c('min', 'max', 'range')
abalone.min.max.range

##      Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## min      15        11      2       0.4           0.2          0.1
## max     163       130     226      565.1         297.6        152.0
## range    148       119     224      564.7         297.4        151.9
##      Shell_weight Rings
## min            0.3     1
## max          201.0    29
## range        200.7    28

# primer cuartil, tercer cuartil y rango intercuartílico.
abalone.Quartiles

##      Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## 25%      90        70     23       88.45         37.25         18.7
## 75%     123       96     33      230.70        100.40        50.6
##      Shell_weight Rings
## 25%      26.00     8
## 75%     65.75    11

abalone.IQR

##      Length      Diameter      Height Whole_weight Shucked_weight
## 33.00      26.00      10.00      142.25        63.15
## Viscera_weight      Shell_weight      Rings
## 31.90      39.75      3.00

# varianza, desviación típica y desviación absoluta de la mediana
abalone.var <- abalone %>% select(-Sex) %>% map_dbl(var)
abalone.sd <- abalone %>% select(-Sex) %>% map_dbl(sd)
abalone.mad <- abalone %>% select(-Sex) %>% map_dbl(mad)

abalone.var

##      Length      Diameter      Height Whole_weight Shucked_weight
## 576.66529   393.78328   69.64039   9617.69749   1970.54863
## Viscera_weight      Shell_weight      Rings
## 480.53407   775.20027   10.39564

abalone.sd

##      Length      Diameter      Height Whole_weight Shucked_weight
##
```

```

##      24.013856      19.843973      8.345082      98.069860      44.390862
## Viscera_weight    Shell_weight      Rings
##      21.921087      27.842419      3.224227
abalone.mad

##      Length      Diameter      Height      Whole_weight Shucked_weight
##      23.72160      19.27380      7.41300      105.56112      46.99842
## Viscera_weight    Shell_weight      Rings
##      23.42508      29.50374      2.96520

```

Cabe destacar que el rango de la variable “Height” es en proporción muy superior a las de las otras dos variables, y que esta proporción no se cumple en el rango intercuartílico. Esto es debido al valor extremo que toma uno de los datos en esta variable y que tendremos que ir descartar para su estudio. Por otro lado la varianza de “Whole weight” es muy superior a la varianza del resto de variables, lo cual puede afectar a la varianza al intentar modelar una regresión con respecto a la variable de salida.

Medidas de forma

Como su nombre indica, las medidas de forma nos proporcionan información sobre la forma de la distribución para una variable. Ya sea una medida de asimetría (skewness) o el tipo de pico que presenta (kurtosis).

```

# Medidas de forma
abalone.skewness <- abalone %>% select(-Sex) %>% map_dbl(skewness)
abalone.kurtosis <- abalone %>% select(-Sex) %>% map_dbl(kurtosis)

```

```
abalone.skewness
```

```

##      Length      Diameter      Height      Whole_weight Shucked_weight
##      -0.6407629      -0.6099623      3.1652259      0.5303580      0.7184768
## Viscera_weight    Shell_weight      Rings
##      0.5912427      0.6208574      1.1133541

```

```
abalone.kurtosis
```

```

##      Length      Diameter      Height      Whole_weight Shucked_weight
##      3.066247      2.955858      79.663065      2.975129      3.592971
## Viscera_weight    Shell_weight      Rings
##      3.082646      3.530288      5.326125

```

```

abalone.skewness2 <- abalone[-abalone.outliers,] %>% select(-Sex) %>%
  map_dbl(skewness)
abalone.kurtosis2 <- abalone[-abalone.outliers,] %>% select(-Sex) %>%
  map_dbl(kurtosis)
abalone.skewness2

```

```

##      Length      Diameter      Height      Whole_weight Shucked_weight
##      -0.4948202      -0.4817499      -0.1525791      0.3453351      0.4551165
## Viscera_weight    Shell_weight      Rings
##      0.4546677      0.3423539      0.2690977

```

```
abalone.kurtosis2
```

```

##      Length      Diameter      Height      Whole_weight Shucked_weight
##      2.574933      2.526550      2.525876      2.348173      2.514875
## Viscera_weight    Shell_weight      Rings
##      2.522992      2.462564      2.784052

```

Antes se había mencionado que las variables que tenían skewness positivo o negativo según la media y mediana, y por lo general se ha cumplido menos para la variable “Height” que también muestra un valor

extremo de kurtosis. Esto se ha corregido eliminando los datos atípicos del estudio, que ahora proporciona valores más lógicos y un skewness negativo para las variables de dimensión

Comprobación de normalidad

Aunque para los algoritmos de regresión que vamos a utilizar no sea de extrema importancia que las distribuciones se asemejen a una normal como para ciertos algoritmos de clasificación, se comprobará igualmente debido a que muchos procesos estadísticos asumen que los datos se distribuyen normalmente, como por ejemplo ciertos tests estadísticos requieren que nuestras variables se distribuyan normalmente. Para comprobar la normalidad de nuestros datos, se aplicará el test de Shapiro-Wilk.

```
# Comprobamos normalidad con shapiro  
  
abalone.shapiro <- apply(abalone[-abalone.outliers,-1],2,shapiro.test)  
abalone.shapiro
```

```
## $Length  
##  
## Shapiro-Wilk normality test  
##  
## data: newX[, i]  
## W = 0.97178, p-value < 2.2e-16  
##  
##  
## $Diameter  
##  
## Shapiro-Wilk normality test  
##  
## data: newX[, i]  
## W = 0.97114, p-value < 2.2e-16  
##  
##  
## $Height  
##  
## Shapiro-Wilk normality test  
##  
## data: newX[, i]  
## W = 0.99142, p-value = 2.52e-14  
##  
##  
## $Whole_weight  
##  
## Shapiro-Wilk normality test  
##  
## data: newX[, i]  
## W = 0.97334, p-value < 2.2e-16  
##  
##  
## $Shucked_weight  
##  
## Shapiro-Wilk normality test  
##  
## data: newX[, i]  
## W = 0.96866, p-value < 2.2e-16  
##  
##
```

```

## $Viscera_weight
##
## Shapiro-Wilk normality test
##
## data: newX[, i]
## W = 0.96918, p-value < 2.2e-16
##
##
## $Shell_weight
##
## Shapiro-Wilk normality test
##
## data: newX[, i]
## W = 0.97726, p-value < 2.2e-16
##
##
## $Rings
##
## Shapiro-Wilk normality test
##
## data: newX[, i]
## W = 0.97437, p-value < 2.2e-16

```

Incluso sin considerar los outliers se obtiene un p-valor muy bajo para todas las variables, menor a 0.05, por lo que el test nos informa de que hay suficiente evidencia estadística que nuestras distribuciones son diferentes a una normal.

Gráficas univariadas de los atributos

Para esta sección, se han efectuado tres diagramas para cada variable: un histograma, un diagrama de cajas para visualizar la naturaleza de los datos atípicos y un diagrama de densidad para apreciar mejor la distribución. Cabe destacar que para las variables de tamaño se ha utilizado el color azul, mientras que para las de peso el color verde y la variable de salida en rojo.

Empezaremos por la única variable categórica, la variable “Sex”:

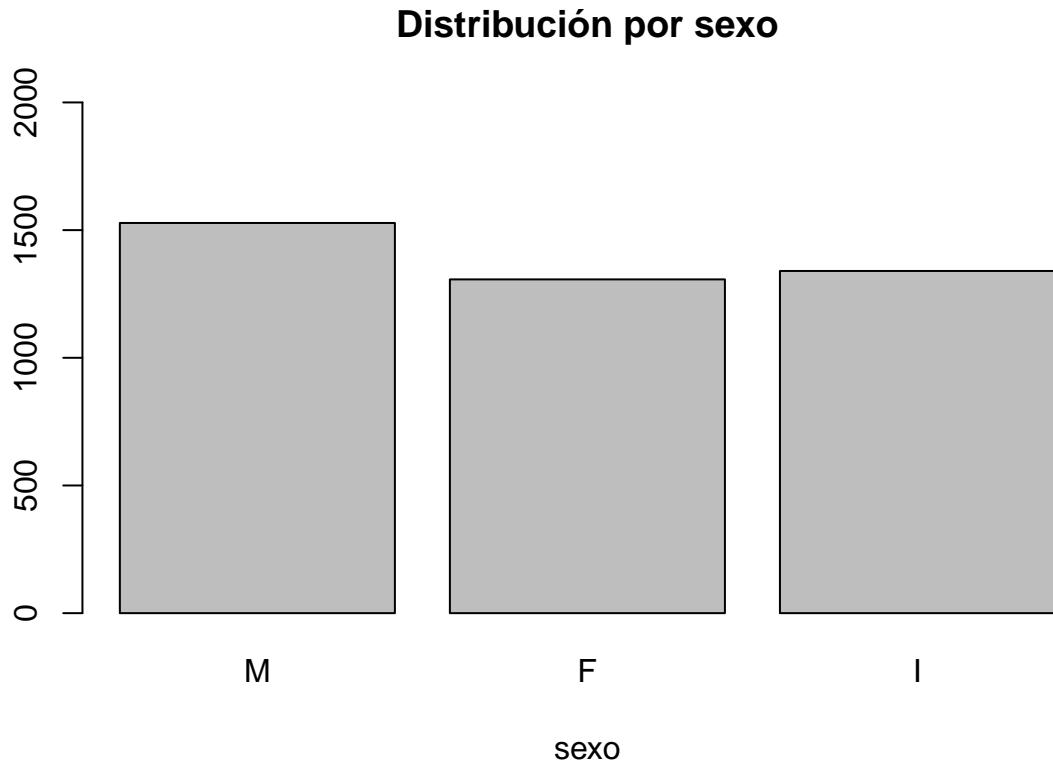
```

counts.sex <- table(abalone$Sex)
counts.sex

##
##      M      F      I
## 1528 1307 1340

barplot(counts.sex, main='Distribución por sexo', xlab='sexo', ylim = c(0,2000))

```

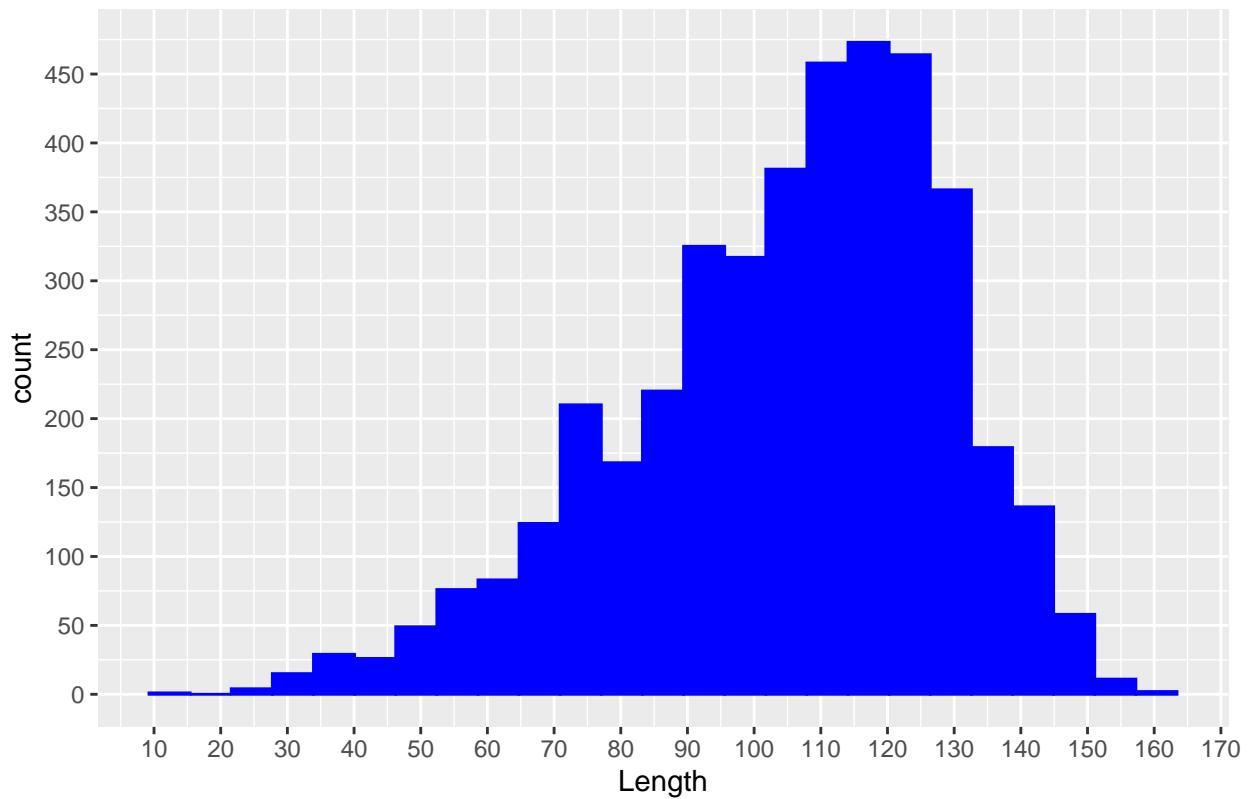


En este gráfico podemos apreciar la cantidad de observaciones de las que disponemos por sexo, siendo la mayoría de los individuos de la clase “male” (1528 observaciones), seguido por la clase “infant” (1340 observaciones), y por último la clase “female” (1528 observaciones). Cabe destacar que con estos valores podemos afirmar que las clases no están muy desbalanceadas.

Para la variable “Length”:

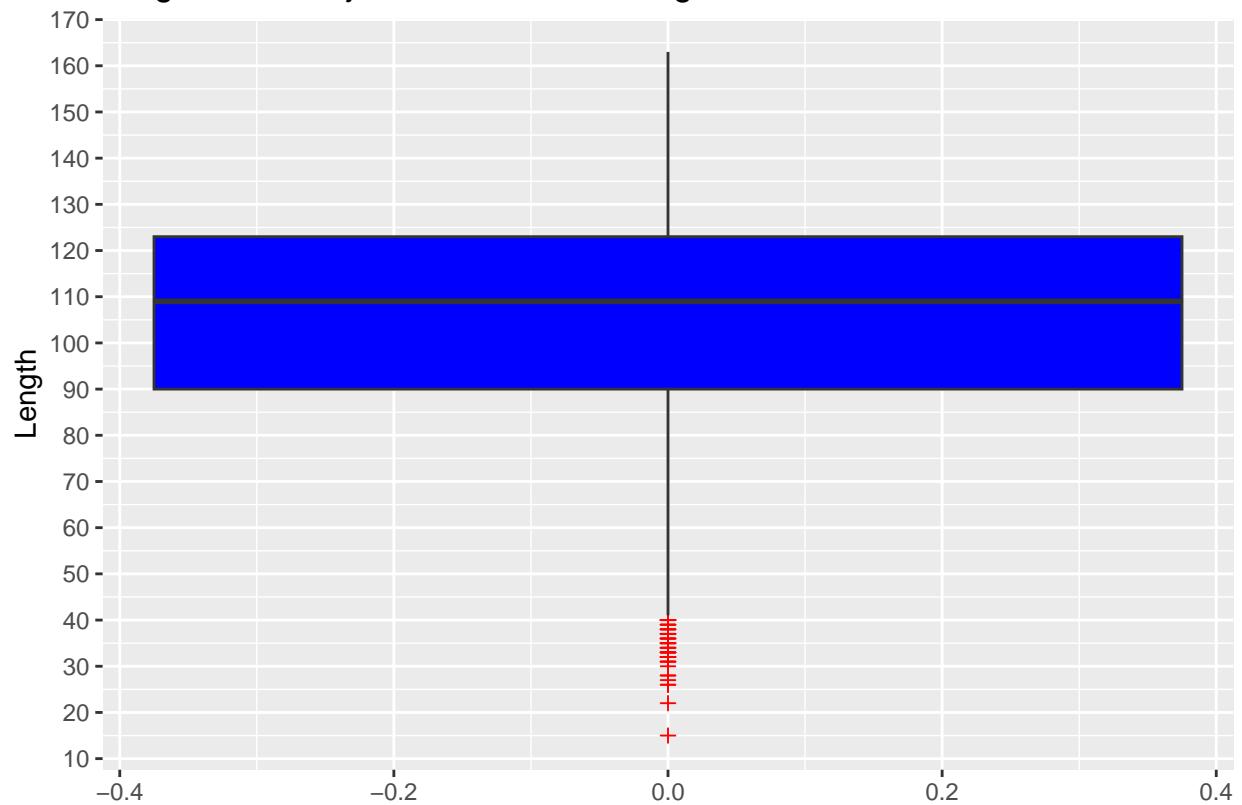
```
# Para Length
ggplot(abalone, aes(x=Length)) +
  geom_histogram(bins=25, color='blue', fill='blue')+
  labs(x='Length', title = 'Histograma de la variable Length')+
  scale_y_continuous(breaks = seq(from=0, to=500, by=50))+
```

Histograma de la variable Length



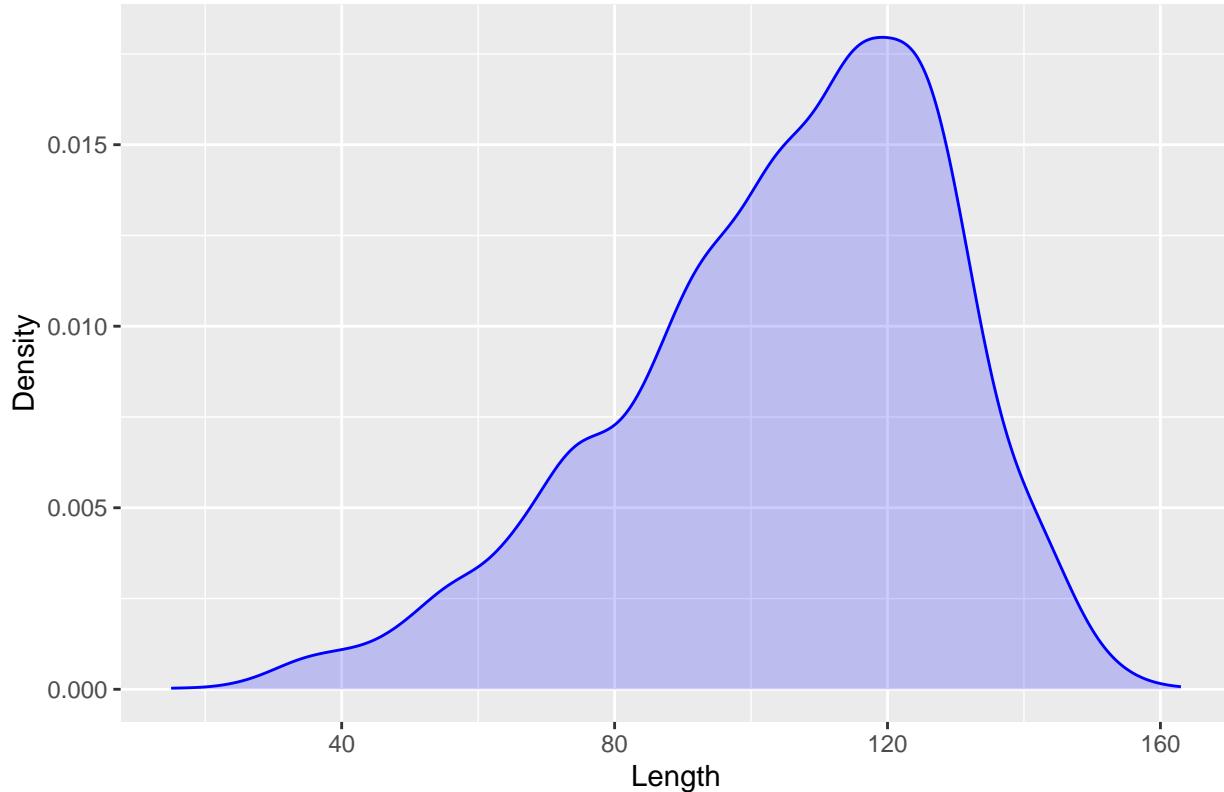
```
ggplot(abalone, aes(y=Length)) +  
  geom_boxplot(fill='blue', outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Length', title = 'Diagrama de cajas de la variable Length')+  
  scale_y_continuous(breaks = seq(from=0, to=170, by=10))
```

Diagrama de cajas de la variable Length



```
ggplot(data=abalone, aes(x=Length, fill="blue"))+  
  geom_density(stat="density", alpha=I(0.2), color= 'blue', fill='blue') +  
  xlab("Length") + ylab("Density") +  
  ggtitle("Curva de densidad de Length")
```

Curva de densidad de Length



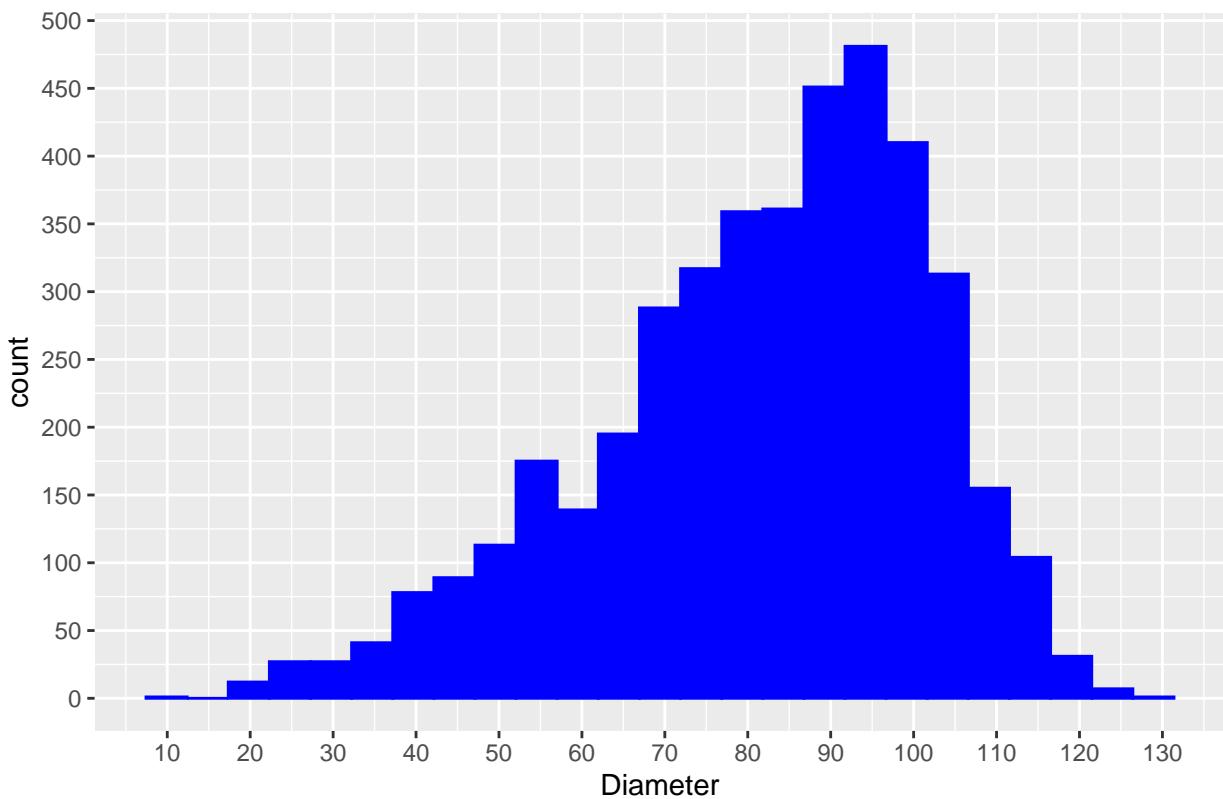
Como se ha mencionado anteriormente, se puede apreciar un “skewness” negativo, es decir la cola más larga de la distribución se encuentra a la izquierda con respecto al centro. También cabe destacar que todos los outliers de esta variable toman valores pequeños estando por debajo del umbral.

Para la variable “Diameter”:

```
# Para Diameter
ggplot(abalone, aes(x=Diameter)) +
  geom_histogram(bins=25, color='blue', fill='blue')+
  labs(x='Diameter', title = 'Histograma de la variable Diameter')+
  scale_y_continuous(breaks = seq(from=0, to=500, by=50))+
```

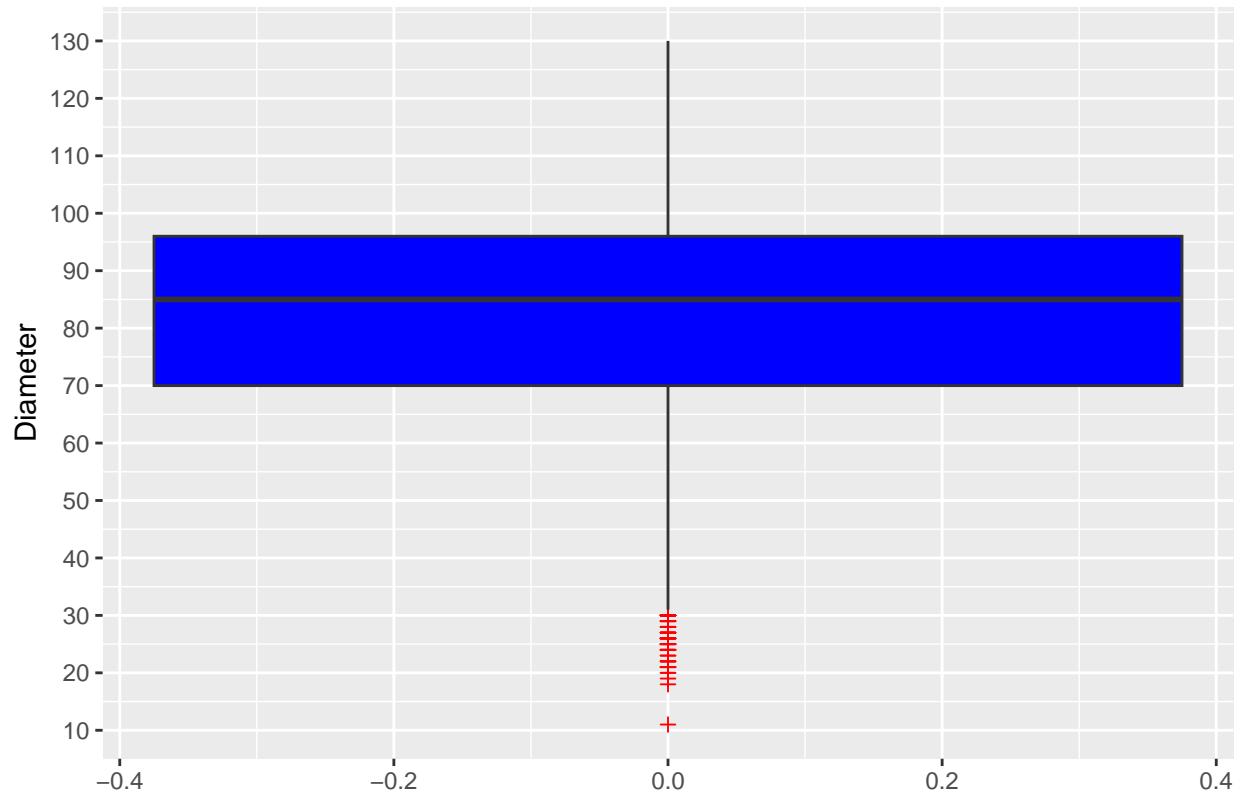
```
scale_x_continuous(breaks = seq(from=0, to=170, by=10))
```

Histograma de la variable Diameter



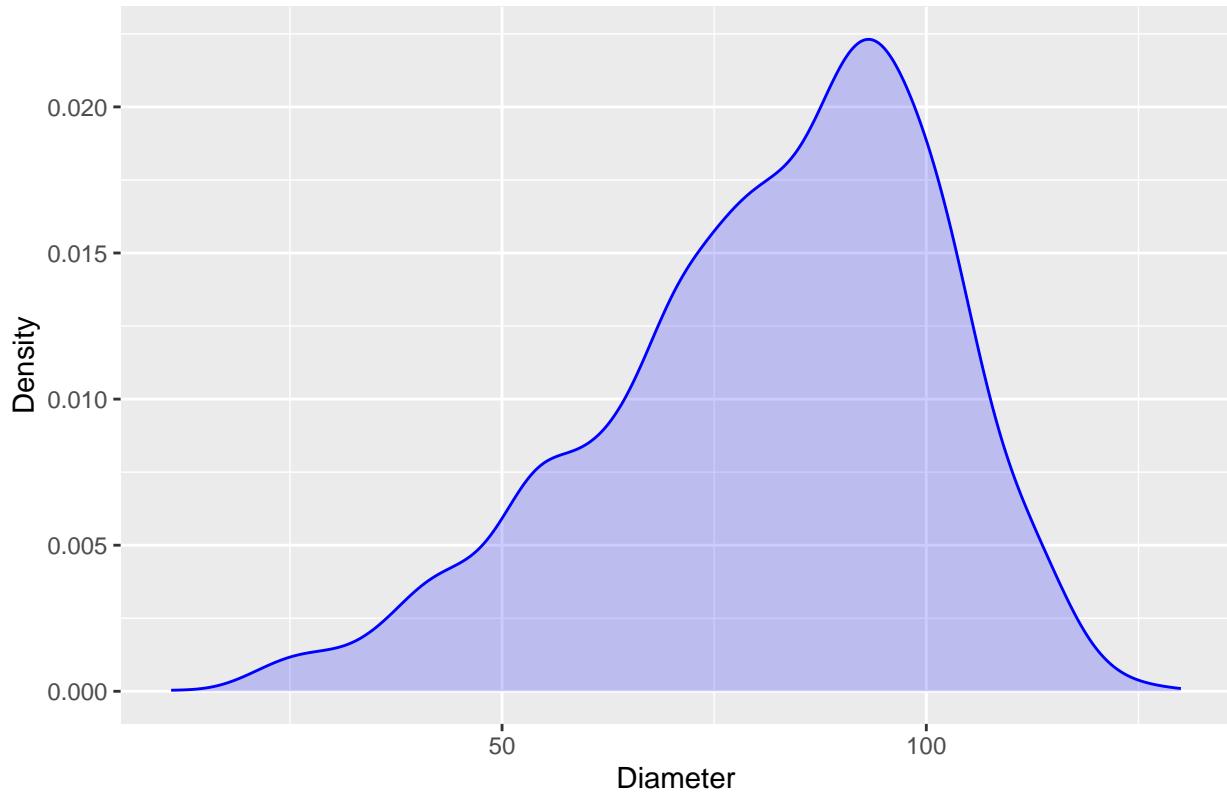
```
ggplot(abalone, aes(y=Diameter)) +  
  geom_boxplot(fill='blue', outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Diameter', title = 'Diagrama de cajas de la variable Diameter')+  
  scale_y_continuous(breaks = seq(from=0, to=170, by=10))
```

Diagrama de cajas de la variable Diameter



```
ggplot(data=abalone, aes(x=Diameter, fill="blue"))+
  geom_density(stat="density", alpha=I(0.2), color= 'blue', fill='blue') +
  xlab("Diameter") + ylab("Density") +
  ggtitle("Curva de densidad de Diameter")
```

Curva de densidad de Diameter



Tenemos una distribución muy similar a la que tenemos en la variable “Length”, con los outliers estando por debajo también. Esto me lleva a pensar que la forma ovalada que toman los abulones se conserva independientemente del tamaño. Sería interesante estudiar estas dos variables juntas y la correlación que hay entre ambas para confirmarlo.

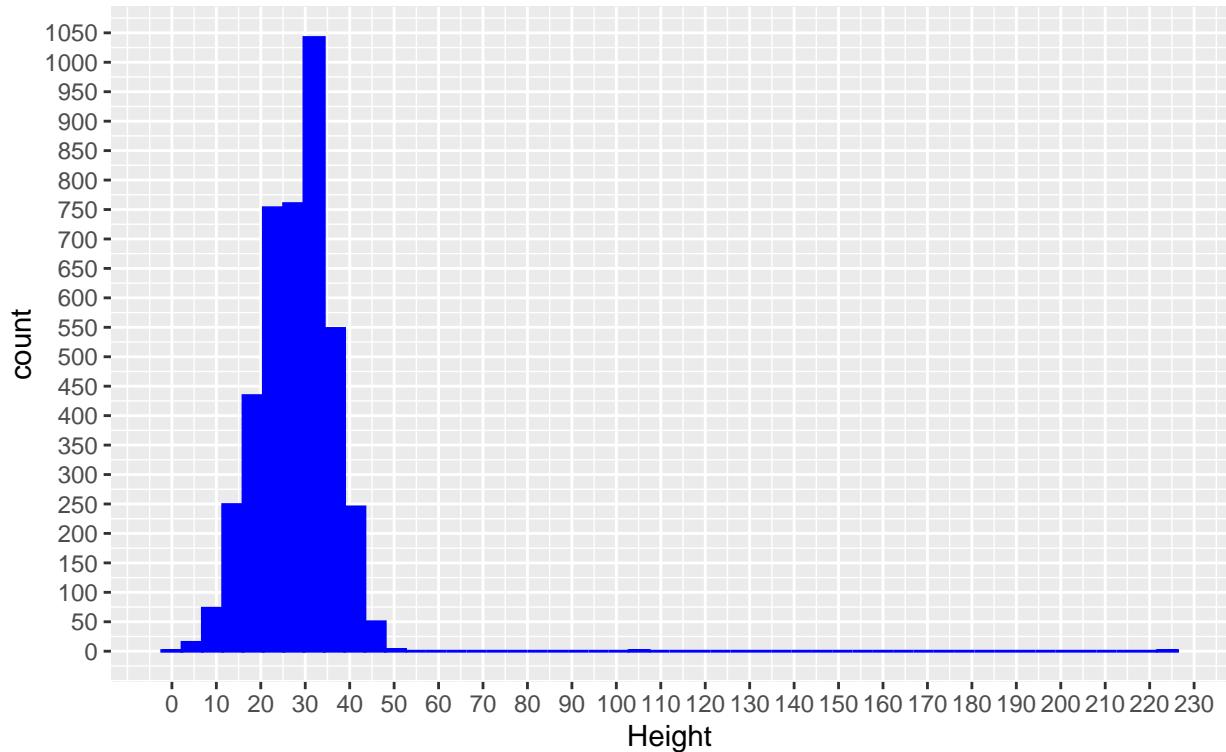
Para la variable “Height”:

```
# Para Height
ggplot(abalone, aes(x=Height)) +
  geom_histogram(bins=50, color='blue', fill='blue')+
  labs(x='Height', title = 'Histograma de la variable Height',
       subtitle = 'Con los outliers de Height')+
  scale_y_continuous(breaks = seq(from=0, to=1200, by=50))+
```

```
scale_x_continuous(breaks = seq(from=0, to=240, by=10))
```

Histograma de la variable Height

Con los outliers de Height

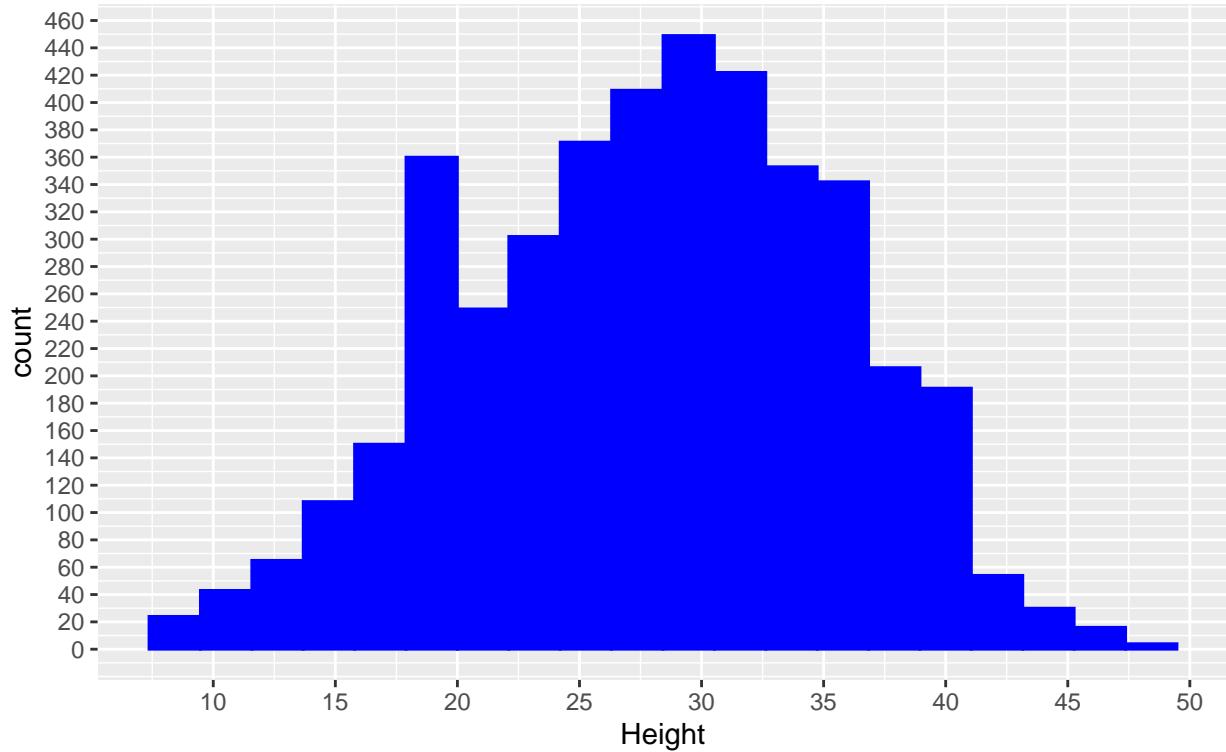


```
# comentar que se ha tenido que coger un mayor número de bins por el outlier
```

```
# Sin outliers:  
ggplot(abalone[-Height.outliers,], aes(x=Height)) +  
  geom_histogram(bins=20, color='blue',fill='blue')+  
  labs(x='Height', title = 'Histograma de la variable Height',  
       subtitle = 'Sin los outliers de Height')+  
  scale_y_continuous(breaks = seq(from=0, to=500, by=20))+  
  scale_x_continuous(breaks = seq(from=0, to=60, by=5))
```

Histograma de la variable Height

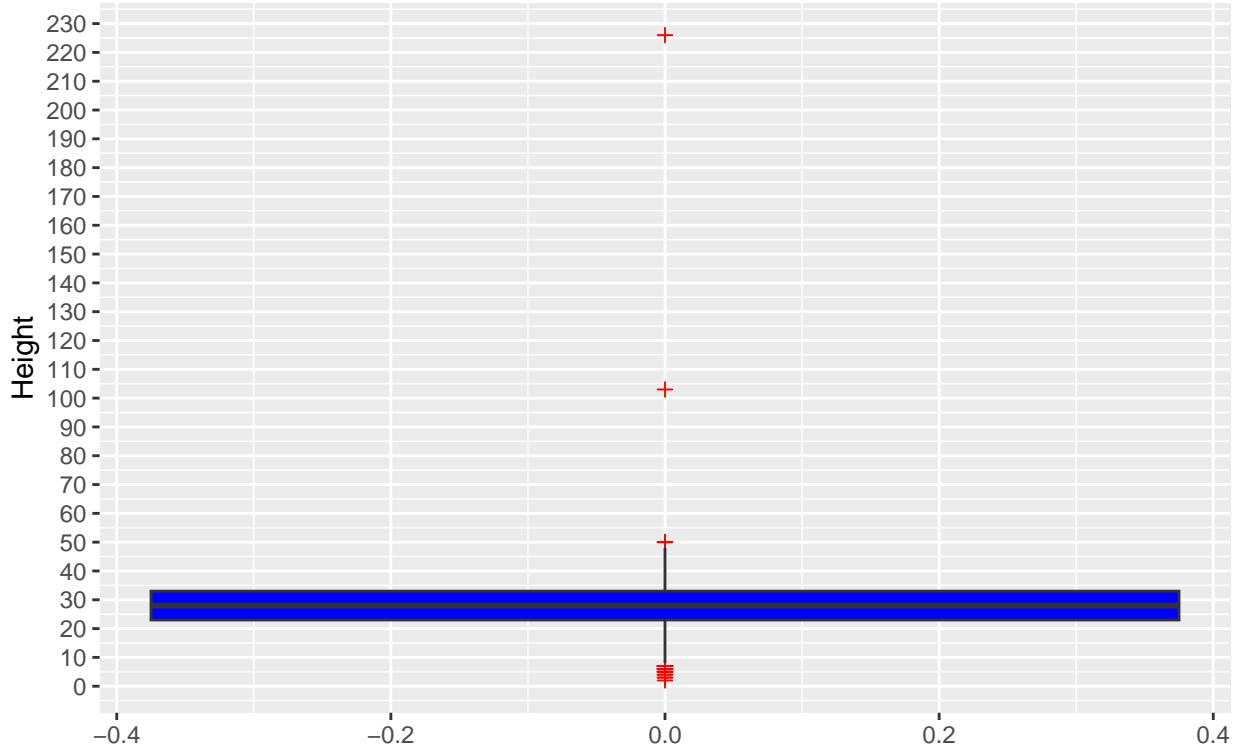
Sin los outliers de Height



```
# boxplot con outliers
ggplot(abalone, aes(y=Height)) +
  geom_boxplot(fill='blue', outlier.colour = 'red', outlier.shape = 3) +
  labs(y='Height', title = 'Diagrama de cajas de la variable Height',
       subtitle = 'Con los outliers de Height') +
  scale_y_continuous(breaks = seq(from=0, to=250, by=10))
```

Diagrama de cajas de la variable Height

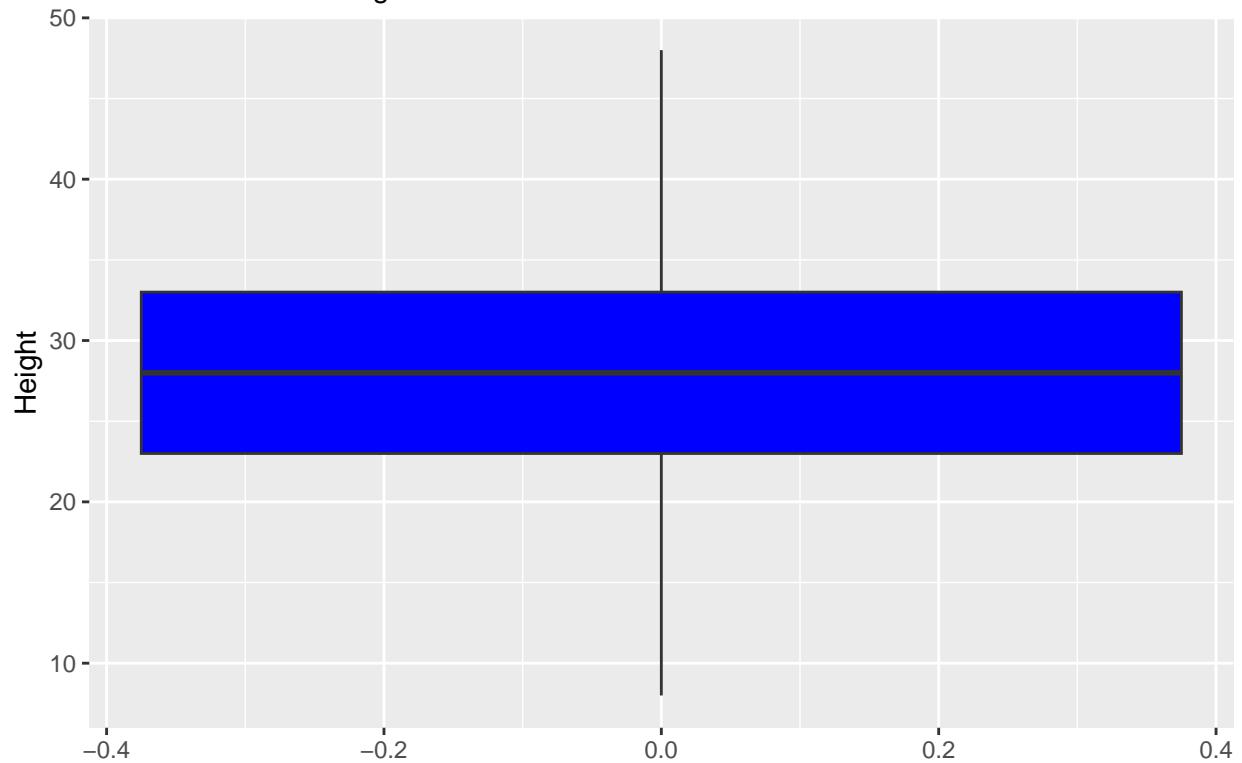
Con los outliers de Height



```
# boxplot sin outliers (quitar los calculados)
ggplot(abalone[-Height.outliers,], aes(y=Height)) +
  geom_boxplot(fill='blue')+
  labs(y='Height', title = 'Diagrama de cajas de la variable Height',
       subtitle = 'Sin los outliers de Height')+
  scale_y_continuous(breaks = seq(from=0, to=250, by=10))
```

Diagrama de cajas de la variable Height

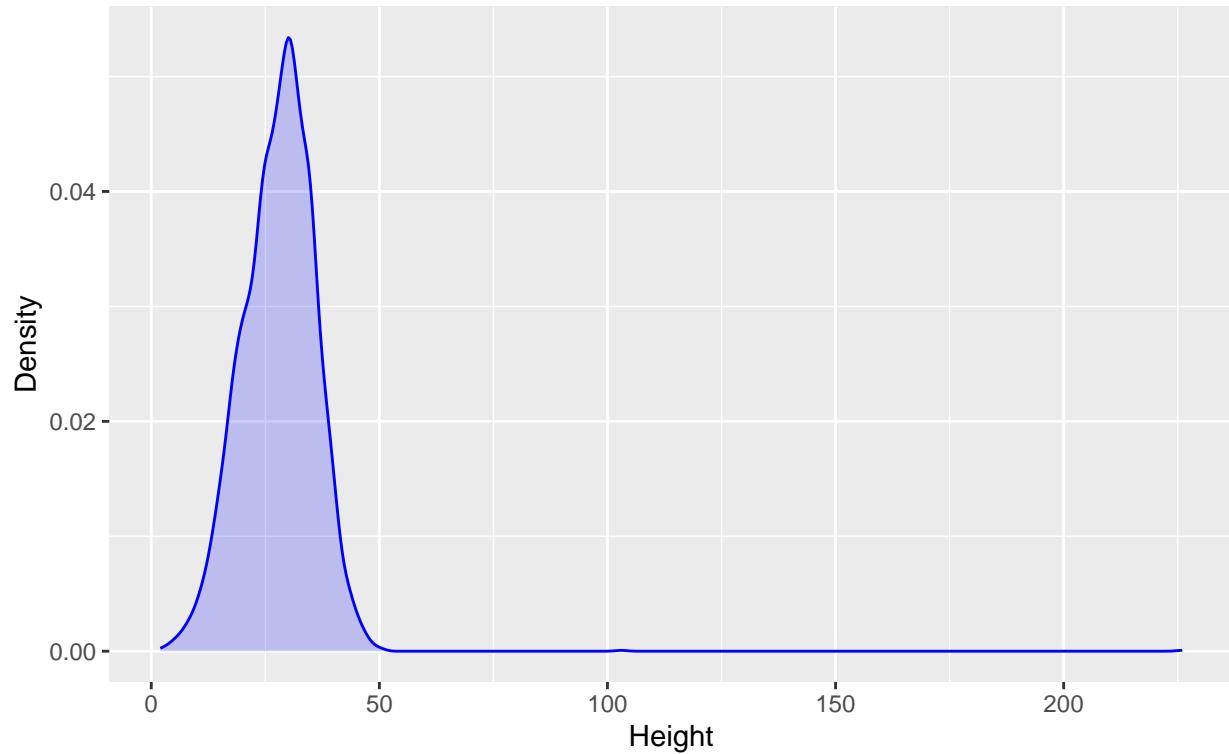
Sin los outliers de Height



```
# curva densidad
ggplot(data=abalone, aes(x=Height, fill="blue"))+
  geom_density(stat="density", alpha=I(0.2), color= 'blue', fill='blue') +
  xlab("Height") + ylab("Density") +
  ggtitle("Curva de densidad de Height", subtitle = 'Con los outliers de Height')
```

Curva de densidad de Height

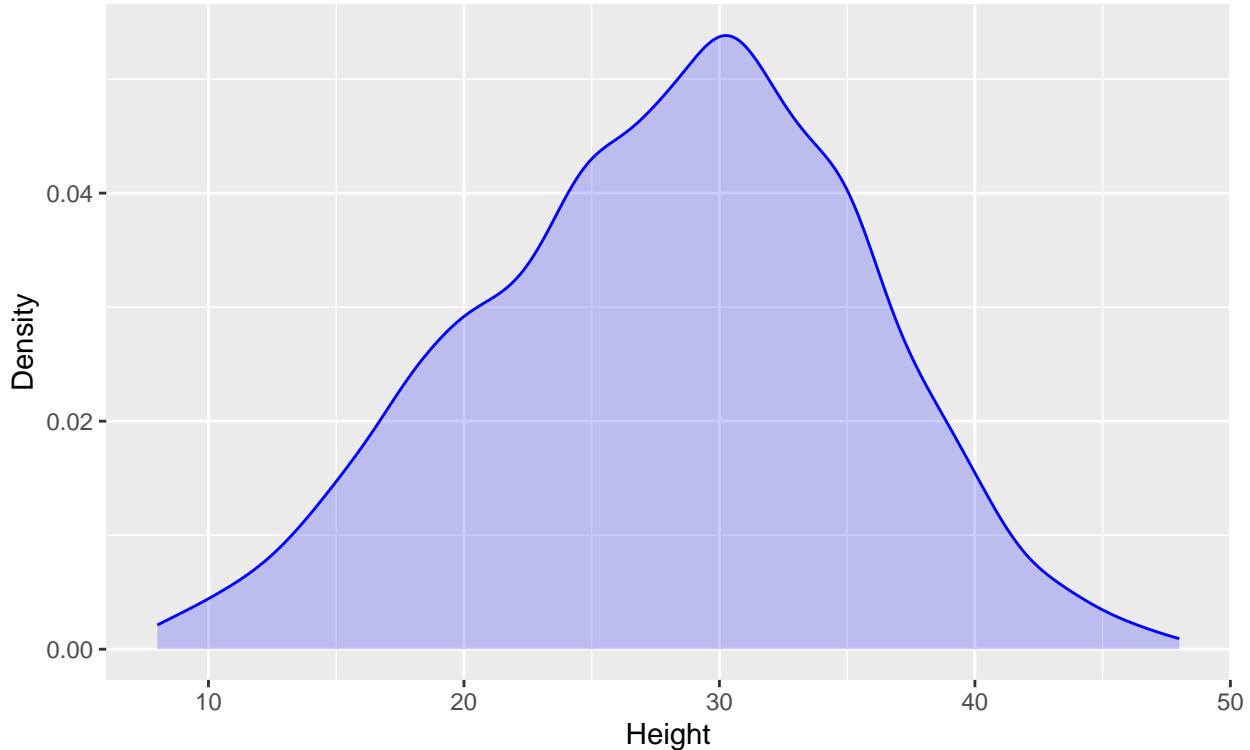
Con los outliers de Height



```
# Sin outliers
ggplot(data=abalone[-Height.outliers,], aes(x=Height, fill="blue"))+
  geom_density(stat="density", alpha=I(0.2), color= 'blue', fill='blue') +
  xlab("Height") + ylab("Density") +
  ggtitle("Curva de densidad de Height", subtitle = 'Sin los outliers de Height')
```

Curva de densidad de Height

Sin los outliers de Height



El valor extremo dificultaba la visualización por lo que para esta variable se ha mostrado también las mismas gráficas sin los outliers, aunque el boxplot si nos confirma de que se trata de un valor demasiado extremo para ser normal. Si buscamos los dos valores:

```
abalone %>% filter(Height>100)
```

```
##   Sex Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## 1   M     141       113    103      442.0       221.5          97.3
## 2   F      91        71    226      118.8       66.4           23.2
##   Shell_weight Rings
## 1         102.4     10
## 2         26.7      8
```

El primer elemento se podría dar, ya que su valor de “Height” no supera al de “Length” y “Diameter”. En el segundo caso, está claro de que se trata de un error en los datos, ya que esta variable supera de manera desproporcionada a las otras dos.

En cuanto a las gráficas, cabe destacar que tenemos un “skewness” negativo, pero no se parece tanto a las otras dos, y es que esta variable se mide teniendo en cuenta la vianda del molusco, por lo que no podemos descartar que la concha no conserve su forma conforme crece en tamaño.

Para la variable “Whole weight”:

```
# Whole_weight
ggplot(abalone, aes(x=Whole_weight)) +
  geom_histogram(bins=24, color='green', fill='green')+
  labs(x='Whole_weight', title = 'Histograma de la variable Whole_weight')+
  scale_y_continuous(breaks = seq(from=0, to=400, by=50))+
```

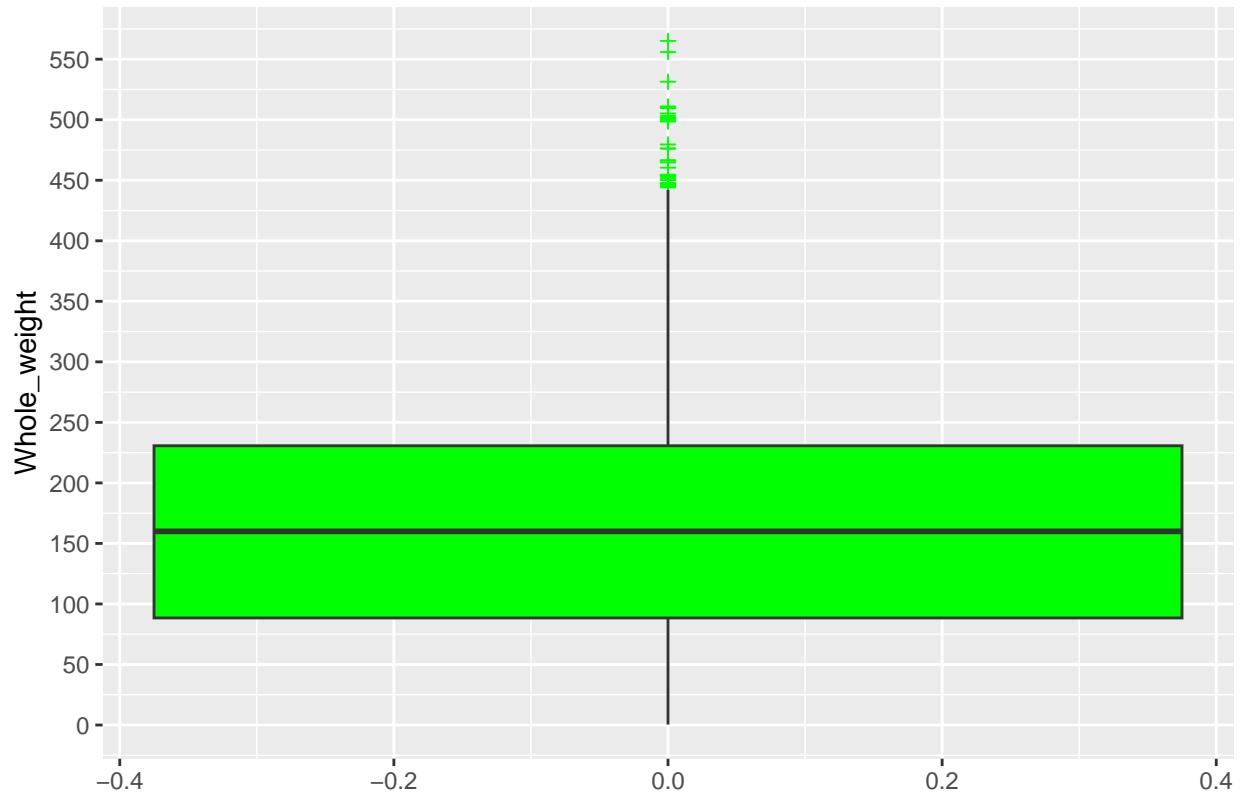
```
scale_x_continuous(breaks = seq(from=0, to=600, by=50))
```

Histograma de la variable Whole_weight



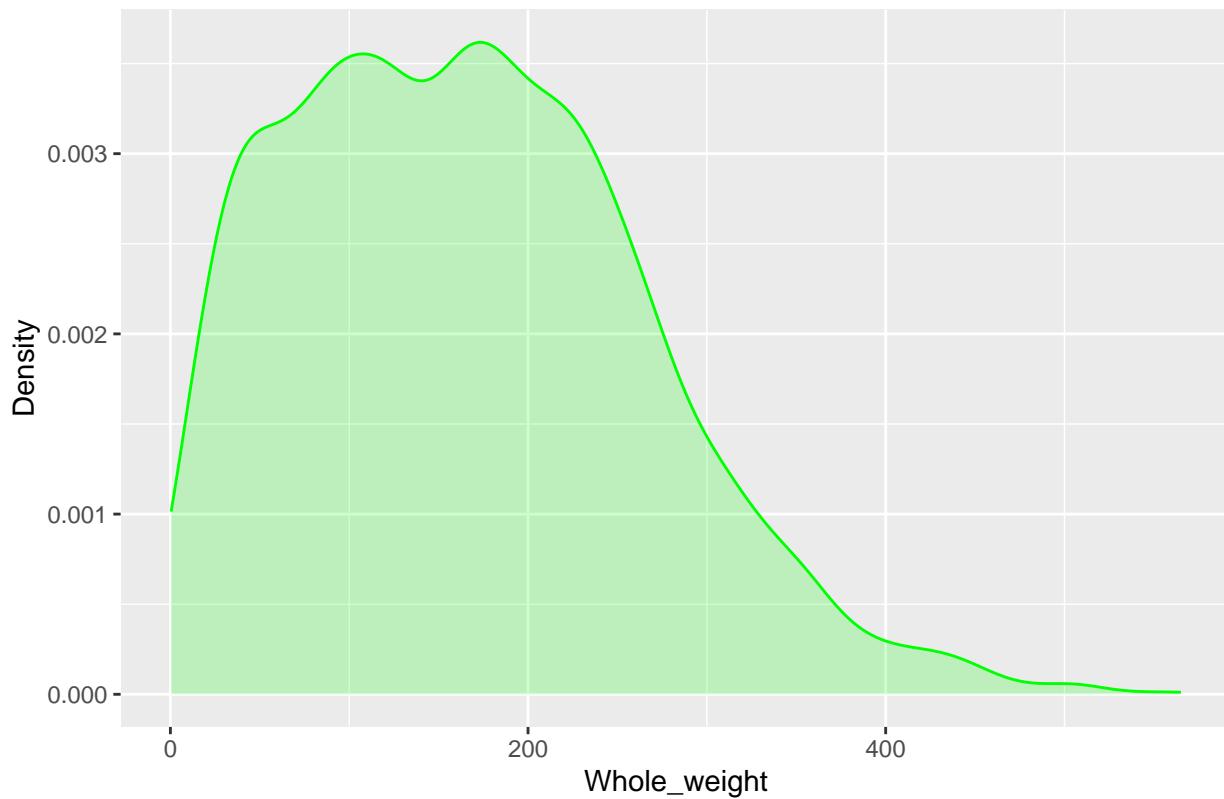
```
ggplot(abalone, aes(y=Whole_weight)) +  
  geom_boxplot(fill='green', outlier.colour = 'green', outlier.shape = 3)+  
  labs(y='Whole_weight', title = 'Diagrama de cajas de la variable Whole_weight')+  
  scale_y_continuous(breaks = seq(from=0, to=600, by=50))
```

Diagrama de cajas de la variable Whole_weight



```
ggplot(data=abalone, aes(x=Whole_weight, fill="green"))+
  geom_density(stat="density", alpha=I(0.2), color= 'green', fill='green') +
  xlab("Whole_weight") + ylab("Density") +
  ggtitle("Curva de densidad de Whole_weight")
```

Curva de densidad de Whole_weight



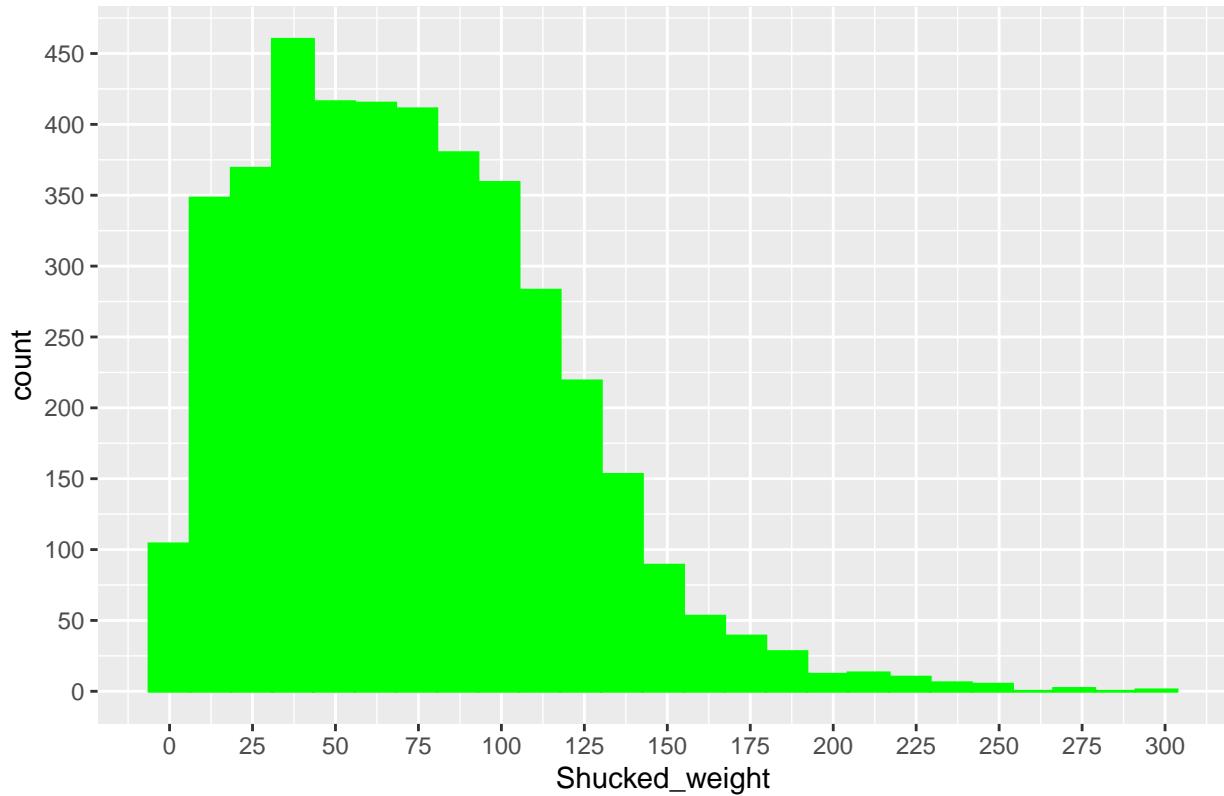
En esta variable de peso tenemos el caso contrario a las dos primeras de tamaño, el “skewness” es positivo, y en este caso los valores atípicos se encuentran por encima del umbral superior. También cabe destacar que en el diagrama de densidad se pueden apreciar dos picos de casi la misma altura (casi una distribución bimodal).

Para “Shucked weight”:

```
# Shucked_weight
ggplot(abalone, aes(x=Shucked_weight)) +
  geom_histogram(bins=25, color='green', fill='green')+
  labs(x='Shucked_weight', title = 'Histograma de la variable Shucked_weight')+
  scale_y_continuous(breaks = seq(from=0, to=500, by=50))+
```

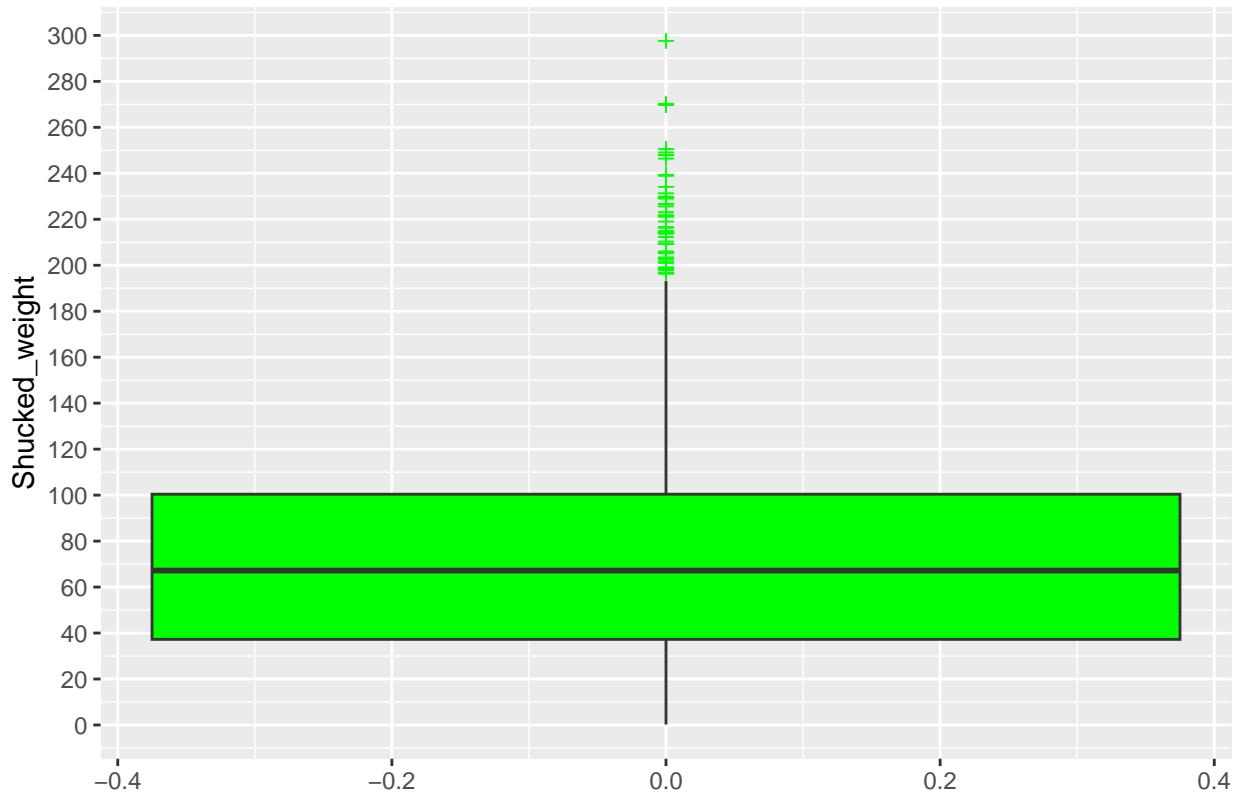
```
scale_x_continuous(breaks = seq(from=0, to=300, by=25))
```

Histograma de la variable Shucked_weight



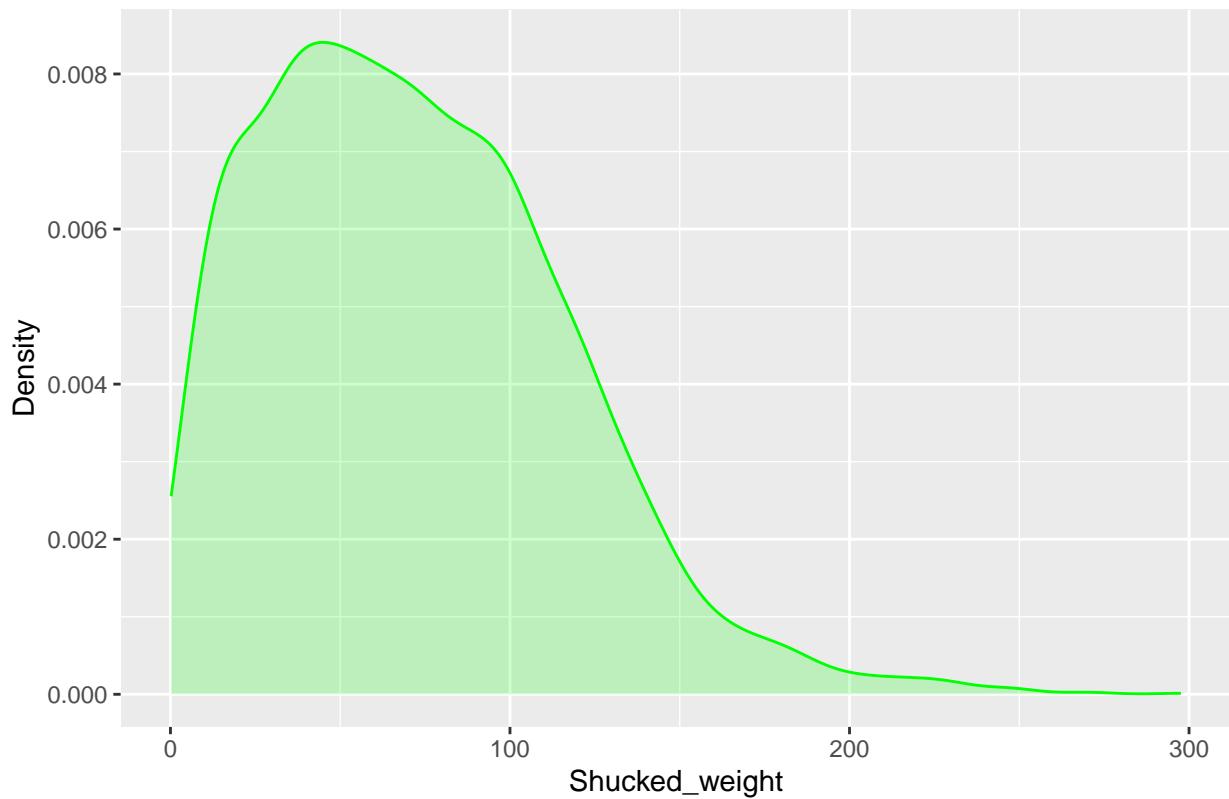
```
ggplot(abalone, aes(y=Shucked_weight)) +  
  geom_boxplot(fill='green', outlier.colour = 'green', outlier.shape = 3)+  
  labs(y='Shucked_weight', title = 'Diagrama de cajas de la variable Shucked_weight')+  
  scale_y_continuous(breaks = seq(from=0, to=300, by=20))
```

Diagrama de cajas de la variable Shucked_weight



```
ggplot(data=abalone, aes(x=Shucked_weight, fill="green"))+
  geom_density(stat="density", alpha=I(0.2), color= 'green', fill='green') +
  xlab("Shucked_weight") + ylab("Density") +
  ggtitle("Curva de densidad de Shucked_weight")
```

Curva de densidad de Shucked_weight

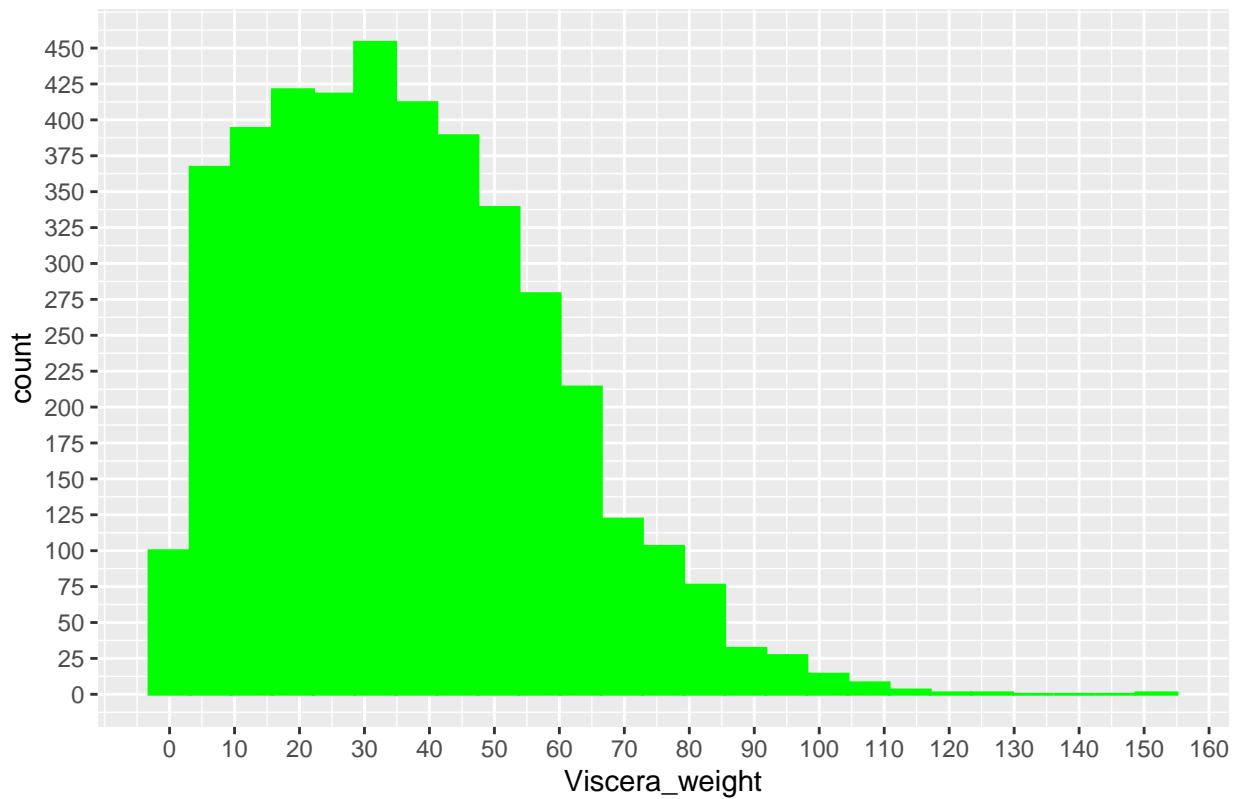


Tenemos una distribución similar a “whole weight” pero con un único pico. Los datos atípicos también se encuentran por encima del umbral superior, y el “skewness” es también positivo.

Para la variable “Viscera weight”

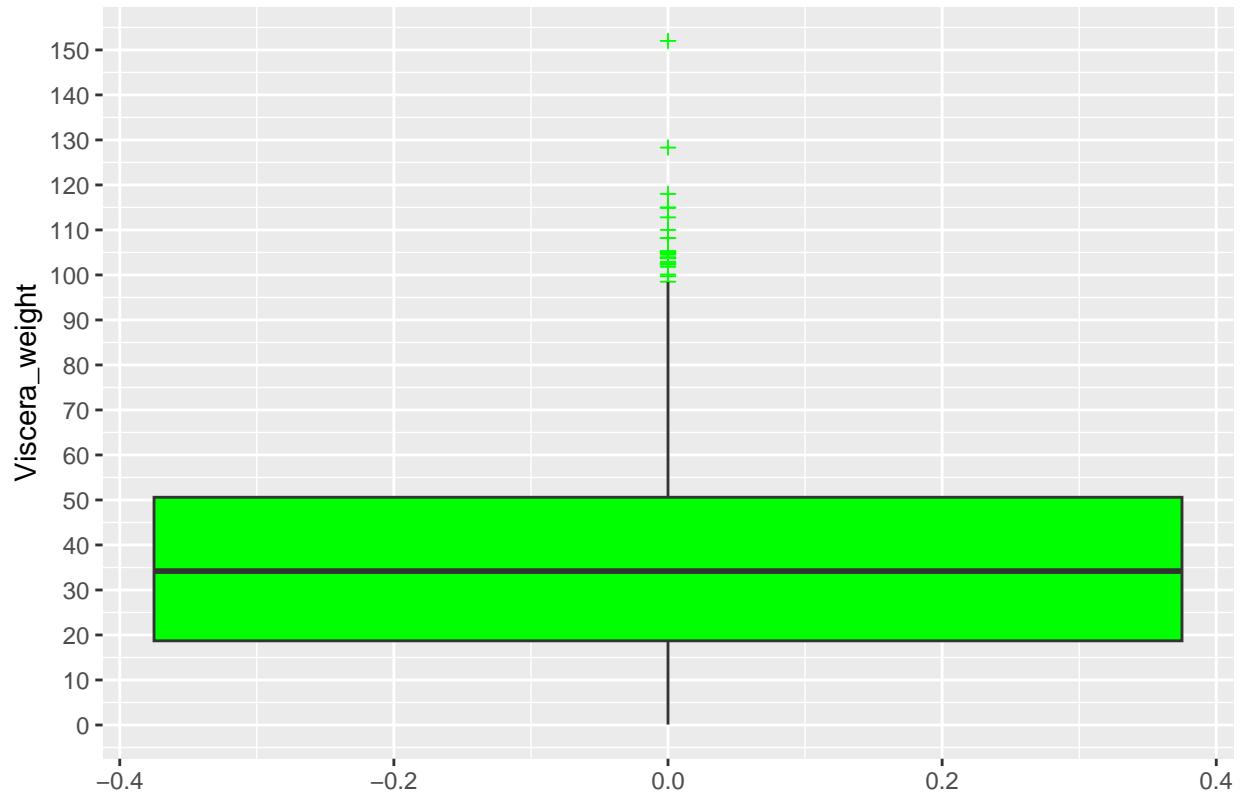
```
# Viscera_weight
ggplot(abalone, aes(x=Viscera_weight)) +
  geom_histogram(bins=25, color='green', fill='green')+
  labs(x='Viscera_weight', title = 'Histograma de la variable Viscera_weight')+
  scale_y_continuous(breaks = seq(from=0, to=450, by=25))+
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))
```

Histograma de la variable Viscera_weight



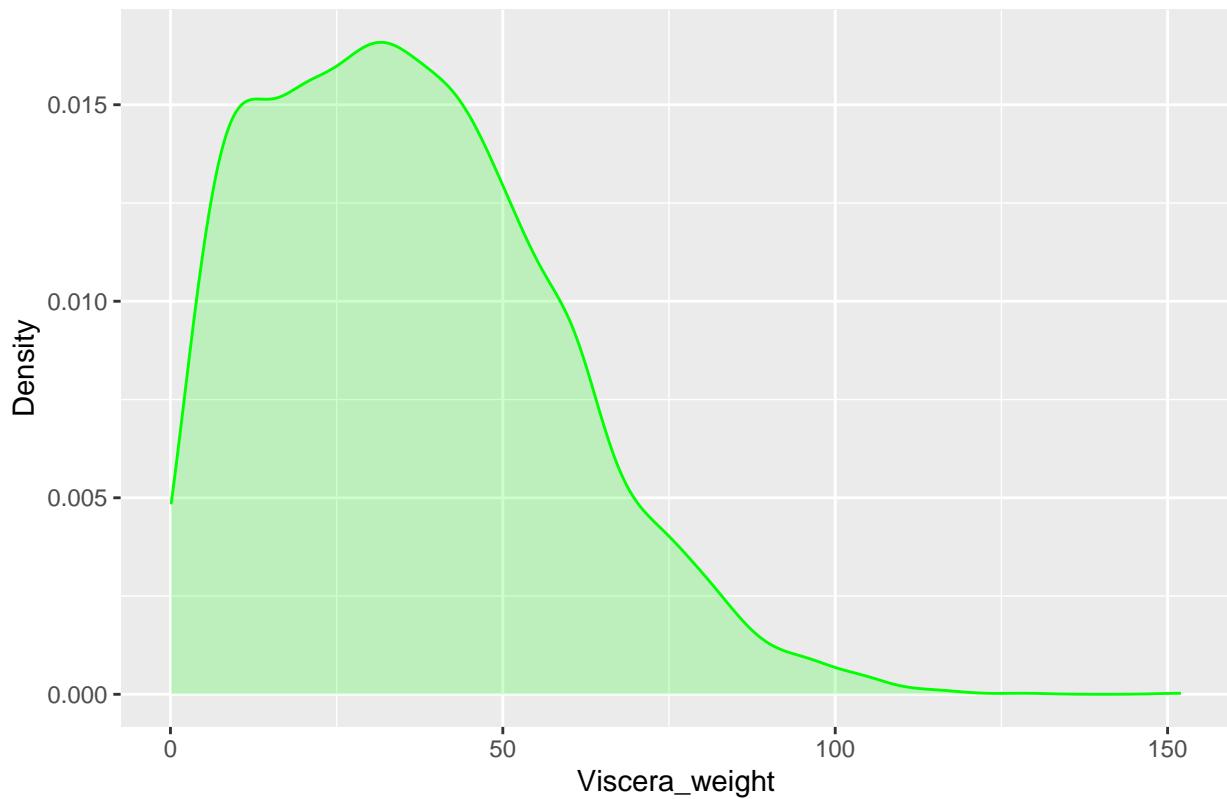
```
ggplot(abalone, aes(y=Viscera_weight)) +  
  geom_boxplot(fill='green', outlier.colour = 'green', outlier.shape = 3)+  
  labs(y='Viscera_weight', title = 'Diagrama de cajas de la variable Viscera_weight')+  
  scale_y_continuous(breaks = seq(from=0, to=160, by=10))
```

Diagrama de cajas de la variable Viscera_weight



```
ggplot(data=abalone, aes(x=Viscera_weight, fill="green"))+  
  geom_density(stat="density", alpha=I(0.2), color= 'green', fill='green') +  
  xlab("Viscera_weight") + ylab("Density") +  
  ggtitle("Curva de densidad de Viscera_weight")
```

Curva de densidad de Viscera_weight



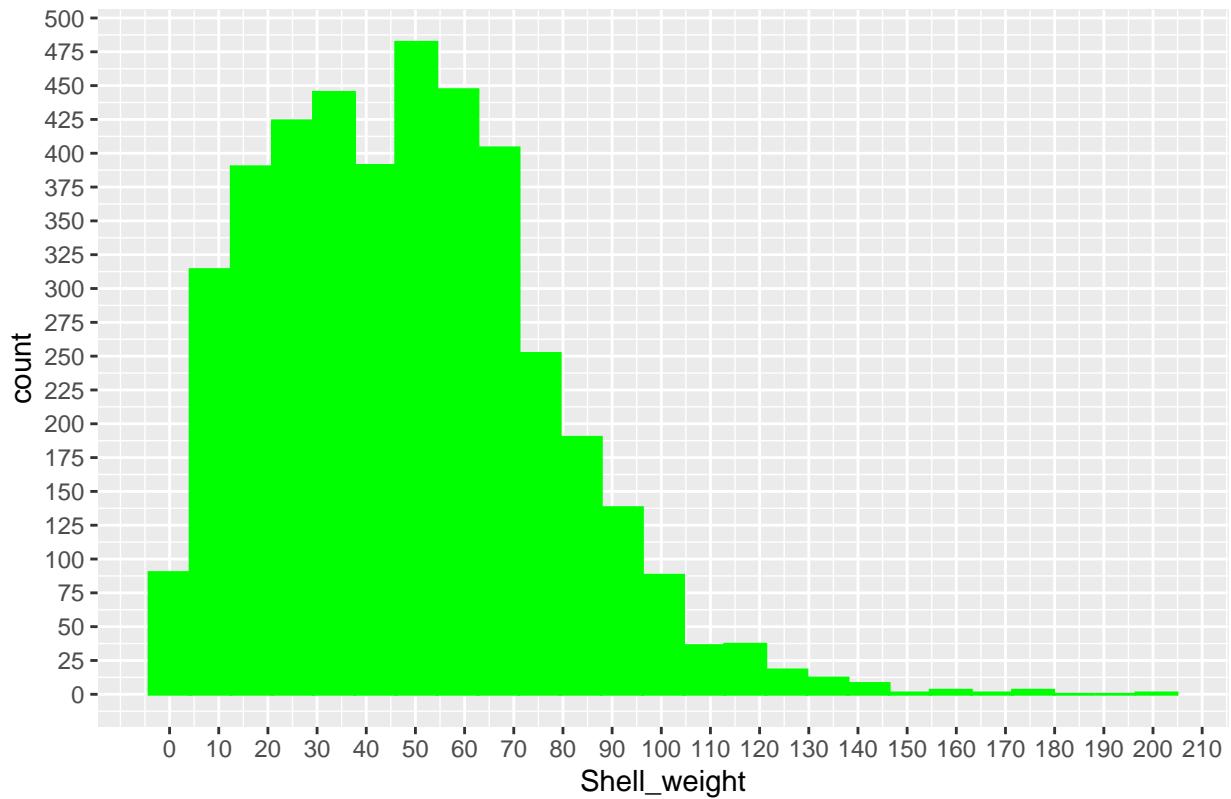
Similar al resto de variables de peso, “skewness” positivo, y datos atípicos con valores altos.

Para las variable “Shell weight”

```
# Shell_weight
ggplot(abalone, aes(x=Shell_weight)) +
  geom_histogram(bins=25, color='green',fill='green')+
  labs(x='Shell_weight', title = 'Histograma de la variable Shell_weight')+
  scale_y_continuous(breaks = seq(from=0, to=500, by=25))+
```

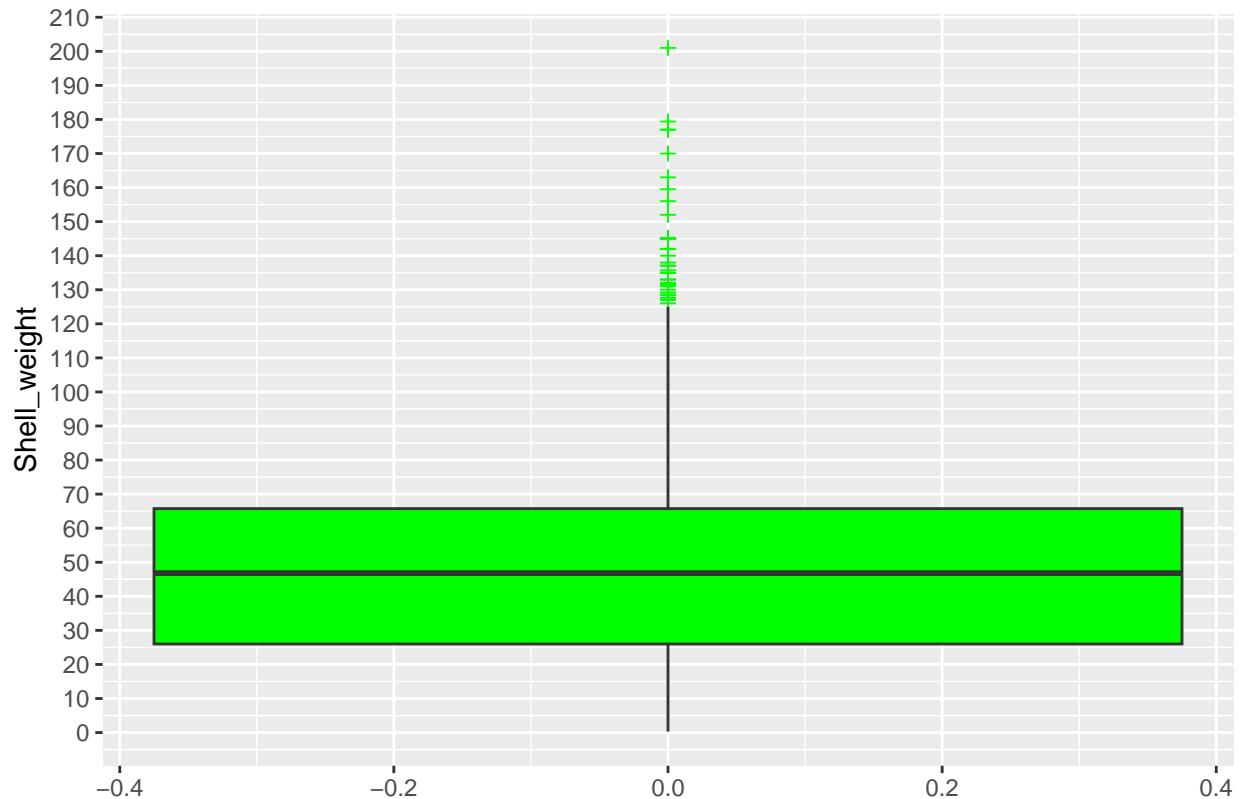
```
scale_x_continuous(breaks = seq(from=0, to=210, by=10))
```

Histograma de la variable Shell_weight



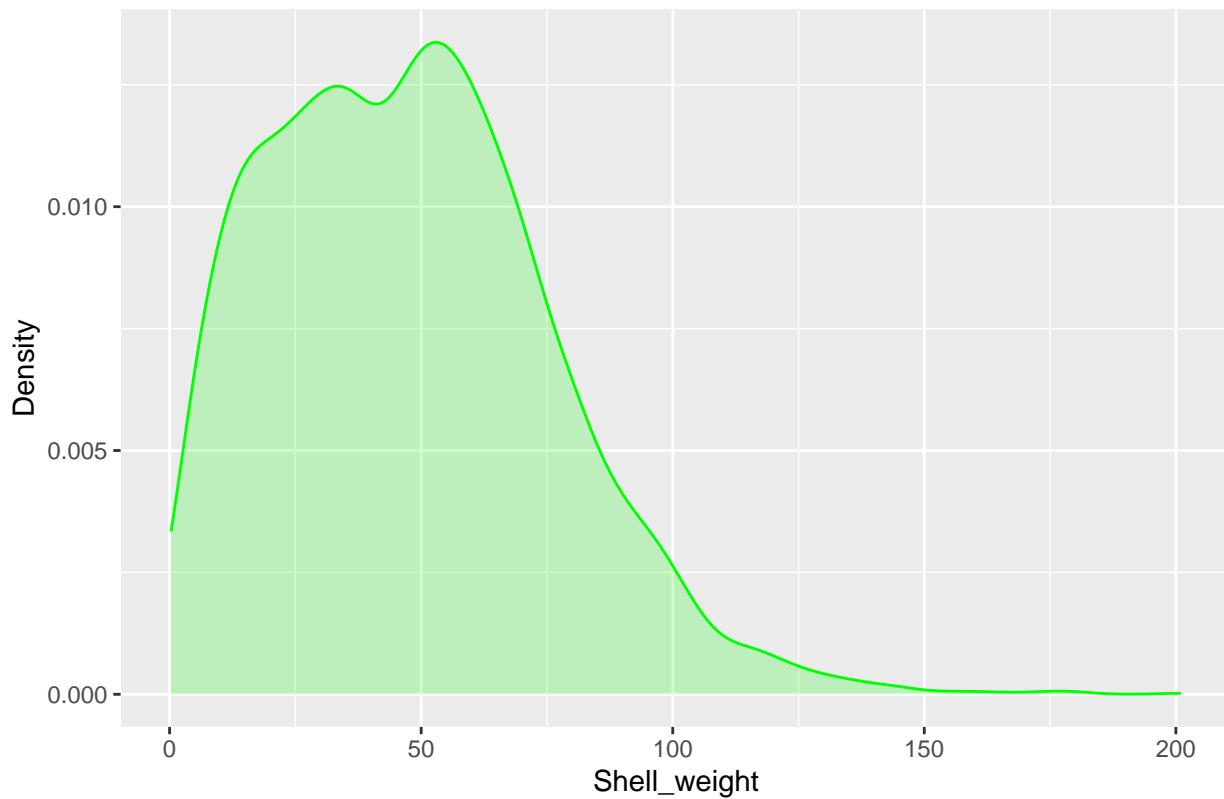
```
ggplot(abalone, aes(y=Shell_weight)) +  
  geom_boxplot(fill='green', outlier.colour = 'green', outlier.shape = 3)+  
  labs(y='Shell_weight', title = 'Diagrama de cajas de la variable Shell_weight')+  
  scale_y_continuous(breaks = seq(from=0, to=210, by=10))
```

Diagrama de cajas de la variable Shell_weight



```
ggplot(data=abalone, aes(x=Shell_weight, fill="green"))+
  geom_density(stat="density", alpha=I(0.2), color= 'green', fill='green') +
  xlab("Shell_weight") + ylab("Density") +
  ggtitle("Curva de densidad de Shell_weight")
```

Curva de densidad de Shell_weight



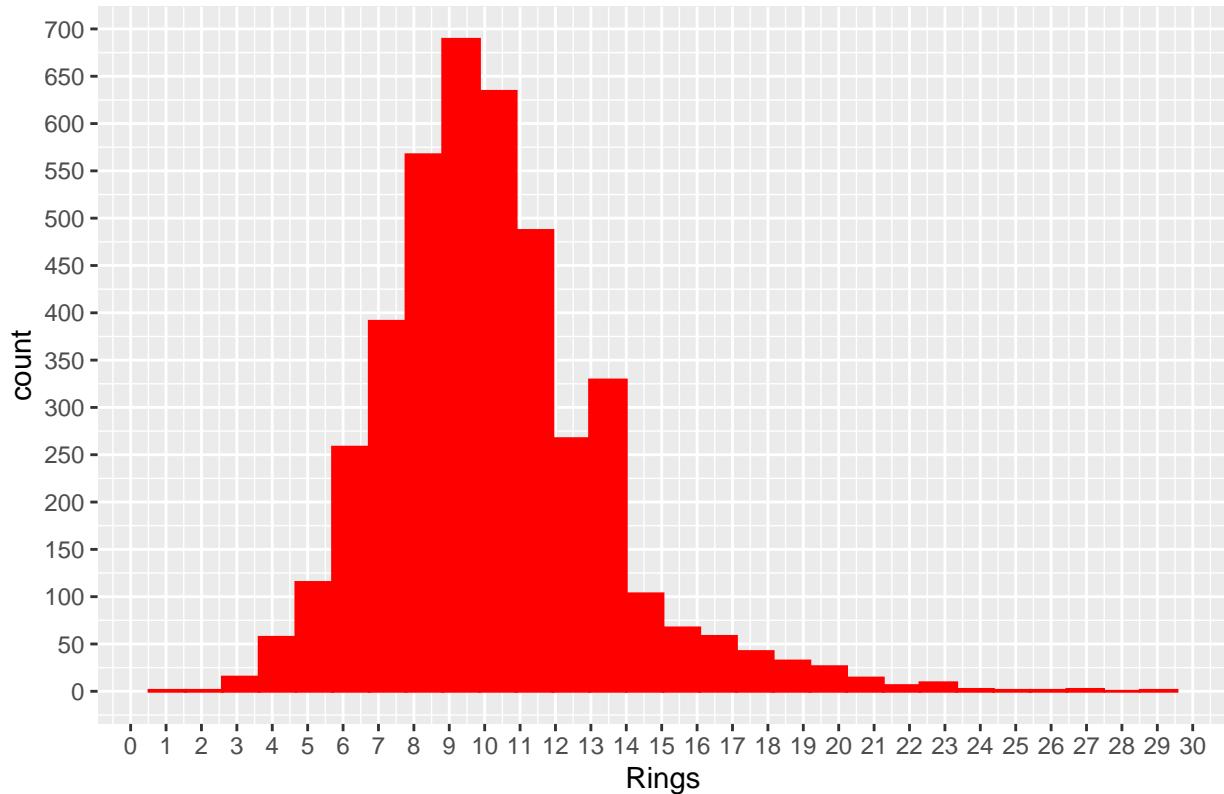
En este caso se obtiene una distribución similar al resto, con outliers con valores altos, pero mirando la función de densidad, parece que tiene dos picos. Fijándonos en todas las distribuciones de las variables de peso, podríamos suponer que la distribución del peso total es aproximadamente la suma de las otras tres, más adelante veremos la correlación que hay entre estas variables.

Por último podemos representar la variable de salida "Rings":

```
# Rings (variable de salida)
ggplot(abalone, aes(x=Rings)) +
  geom_histogram(bins=length(unique(abalone$Rings)), color='Red',fill='Red')+
  labs(x='Rings', title = 'Histograma de la variable Rings')+
  scale_y_continuous(breaks = seq(from=0, to=700, by=50))+
```

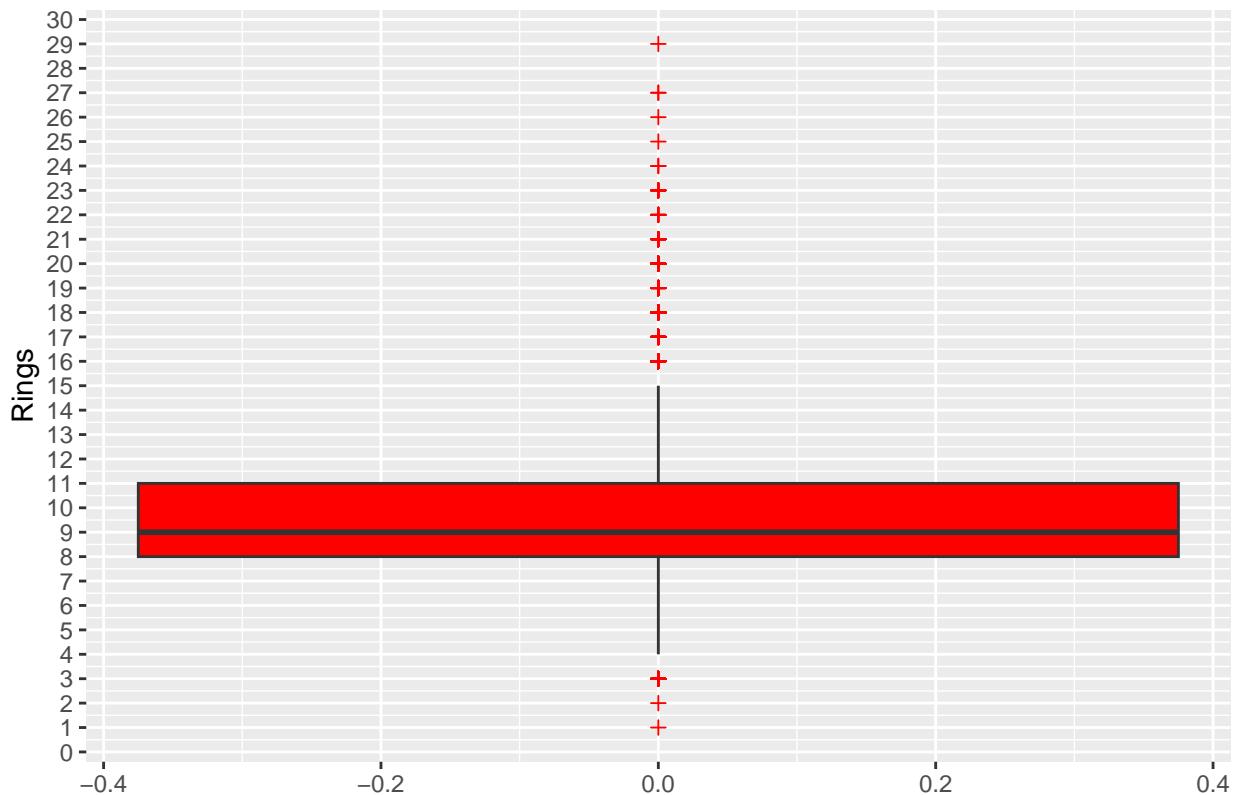
```
scale_x_continuous(breaks = seq(from=0, to=30, by=1))
```

Histograma de la variable Rings



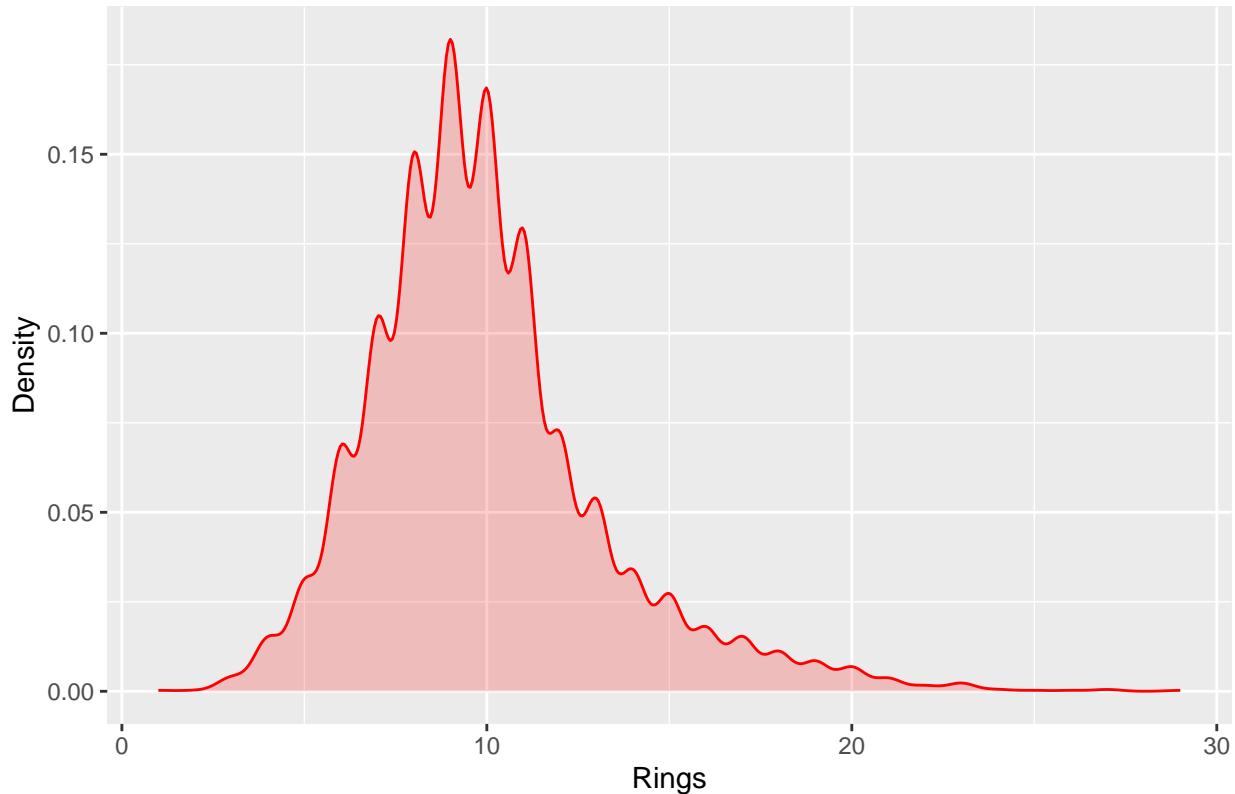
```
ggplot(abalone, aes(y=Rings)) +  
  geom_boxplot(fill='Red', outlier.colour = 'Red', outlier.shape = 3)+  
  labs(y='Rings', title = 'Diagrama de cajas de la variable Rings')+  
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))
```

Diagrama de cajas de la variable Rings



```
ggplot(data=abalone, aes(x=Rings, fill="Red"))+  
  geom_density(stat="density", alpha=I(0.2), color= 'Red', fill='Red') +  
  xlab("Rings") + ylab("Density") +  
  ggtitle("Curva de densidad de Rings")
```

Curva de densidad de Rings



La distribución de la variable de salida es la más puntiaguda (con mayor medida del coeficiente de kurtosis) y lo que quiere decir que los valores de los individuos se encuentran más centrados. Su función de densidad tiene forma de sierra debido a que toma valores enteros (valores discretos).

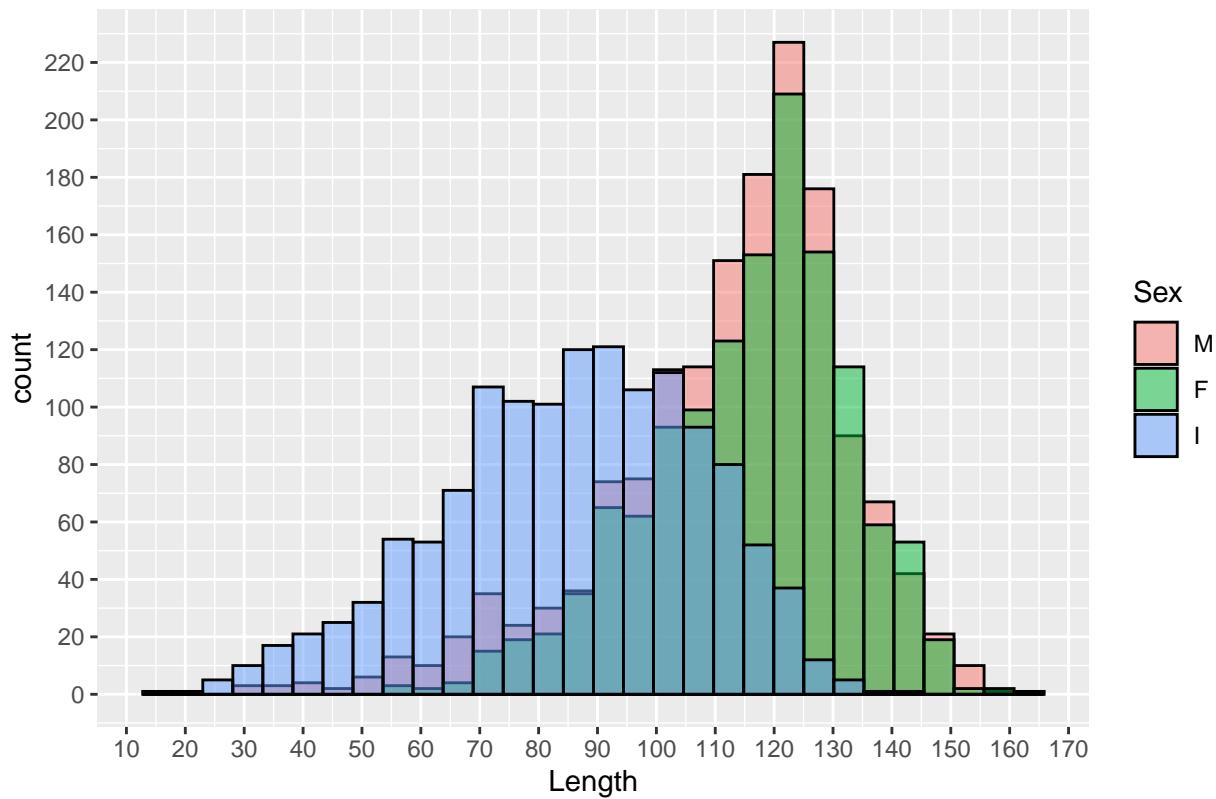
Gráficas bivariadas de los atributos

Al ser la variable “Sex” la única categórica, la analizaremos la primera con respecto al resto de variables, y así podremos comprobar algunas de las hipótesis planteadas. Para representar esta variable frente a las demás se han hecho uso de diagramas de cajas e histogramas.

Empezamos por analizar el par “Sex-Length”:

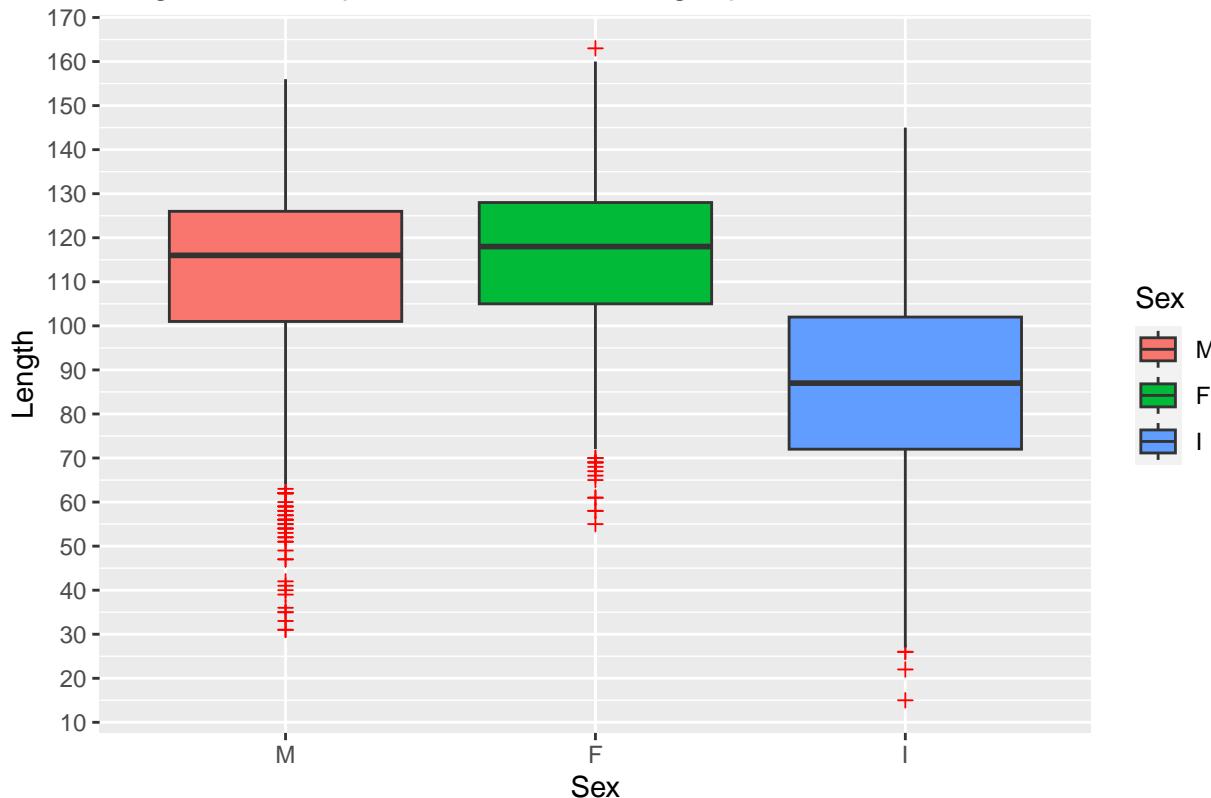
```
# sex-Length  
  
ggplot(abalone, aes(x=Length, fill=Sex)) +  
  geom_histogram(bins=30, color='black', alpha=0.5, position='identity') +  
  labs(x='Length', title = 'Histograma de Length en función del sexo') +  
  scale_y_continuous(breaks = seq(from=0, to=240, by=20)) +  
  scale_x_continuous(breaks = seq(from=0, to=170, by=10))
```

Histograma de Length en función del sexo



```
ggplot(abalone, aes(y=Length, x=Sex)) +  
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Length', title = 'Diagrama de cajas de la variable Length por sexo')+  
  scale_y_continuous(breaks = seq(from=0, to=170, by=10))
```

Diagrama de cajas de la variable Length por sexo

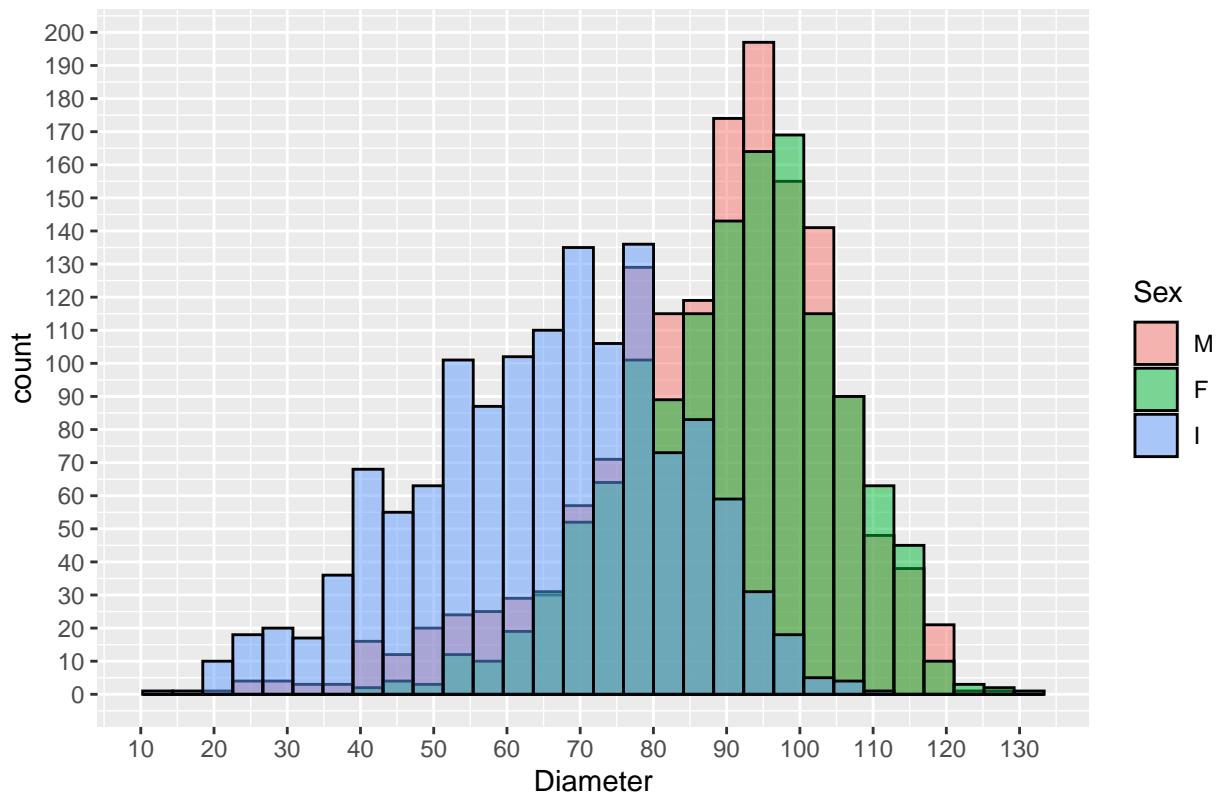


Sorprendentemente no se puede apreciar mucha diferencia en la longitud entre los dos sexos, como mucho se puede observar mayor dispersión en los datos para los machos que para las hembras (mayor número de outliers también). Donde si hay una diferencia es en la categoría “infant” cuya distribución toma valores menores con cierto solapamiento con las distribuciones de las otras dos categorías. Este fenómeno ya se comentó anteriormente y se dieron algunas posibles explicaciones de lo que podría estar ocurriendo.

Analizamos “Sex-Diameter”:

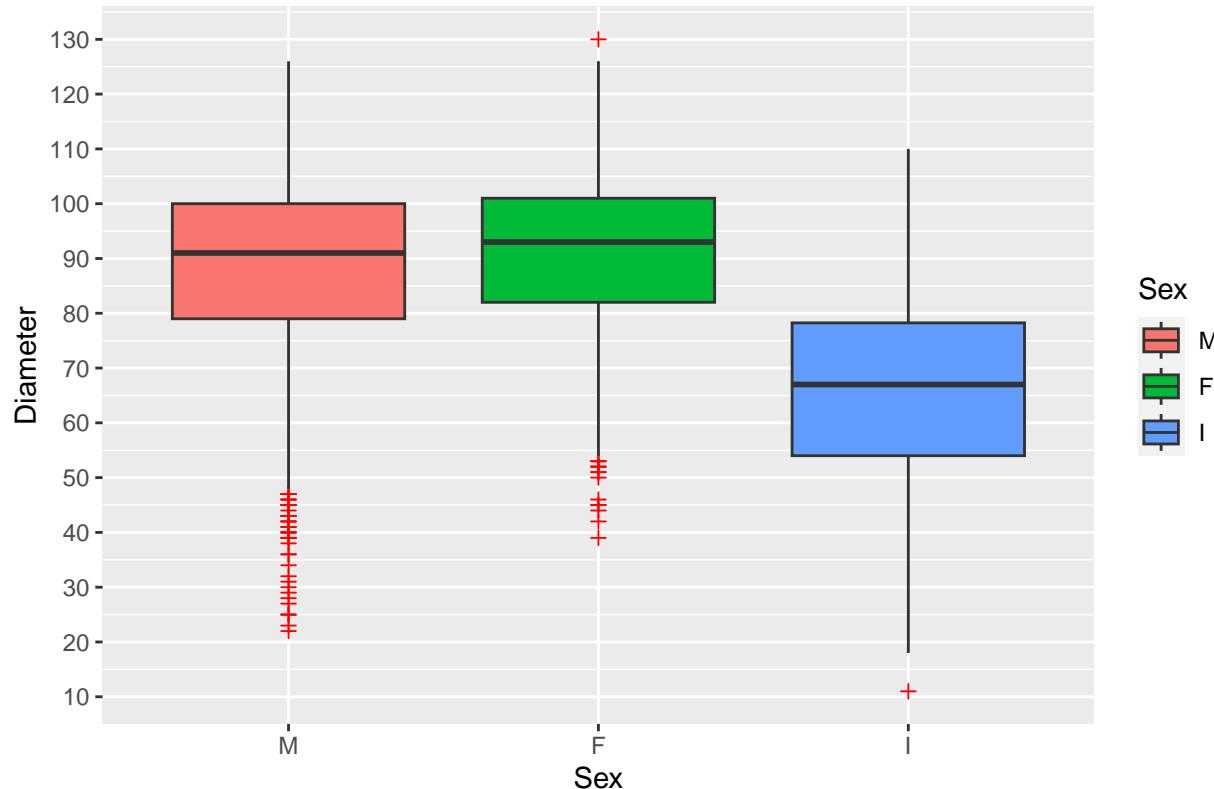
```
#sex-Diameter
ggplot(abalone, aes(x=Diameter, fill=Sex)) +
  geom_histogram(bins=30, color='black', alpha=0.5, position='identity')+
  labs(x='Diameter', title = 'Histograma de Diameter en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=200, by=10))+
```

Histograma de Diameter en función del sexo



```
ggplot(abalone, aes(y=Diameter, x=Sex)) +  
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Diameter', title = 'Diagrama de cajas de la variable Diameter por sexo')+  
  scale_y_continuous(breaks = seq(from=0, to=170, by=10))
```

Diagrama de cajas de la variable Diameter por sexo

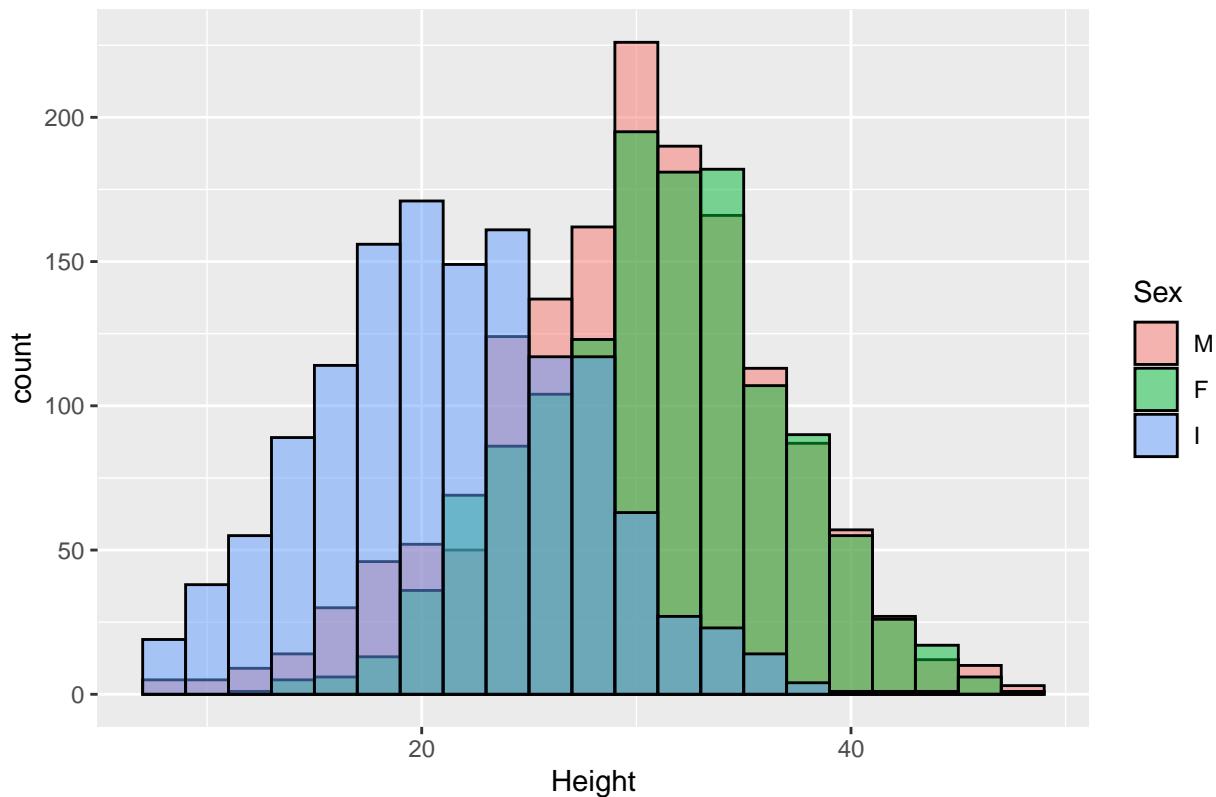


De la misma forma, no se aprecia mucha diferencia en esta dimensión tampoco entre los dos sexos, hay que remarcar que la categoría de “male” tiene mayores frecuencias para algunos intervalos mayores que “female”, pero esto se podría deber a que es la clase con el mayor número de individuos, mientras que “female” es la que menos individuos tiene de las tres. En este caso se aprecia lo mismo con respecto a la clase “infant” que en el gráfico anterior. Cabe destacar que esta clase es la que menos outliers tiene de las tres para estas variables.

Para “Sex-Height”:

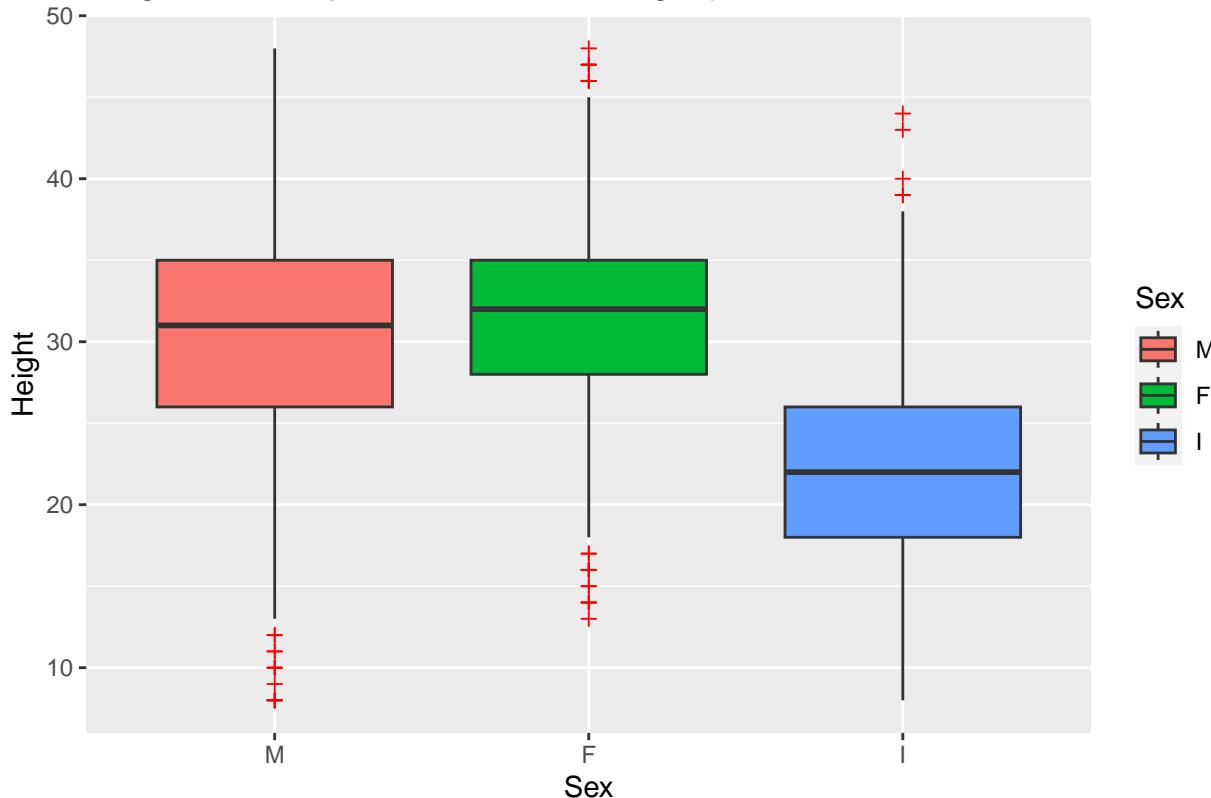
```
#sex-Height
ggplot(abalone[-Height.outliers,], aes(x=Height, fill=Sex)) +
  geom_histogram(bins=21, color='black', alpha=0.5, position='identity')+
  labs(x='Height', title = 'Histograma de Height en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=1500, by=50))+
```

Histograma de Height en función del sexo



```
ggplot(abalone[-Height.outliers,], aes(y=Height, x=Sex)) +  
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3) +  
  labs(y='Height', title = 'Diagrama de cajas de la variable Height por sexo') +  
  scale_y_continuous(breaks = seq(from=0, to=170, by=10))
```

Diagrama de cajas de la variable Height por sexo

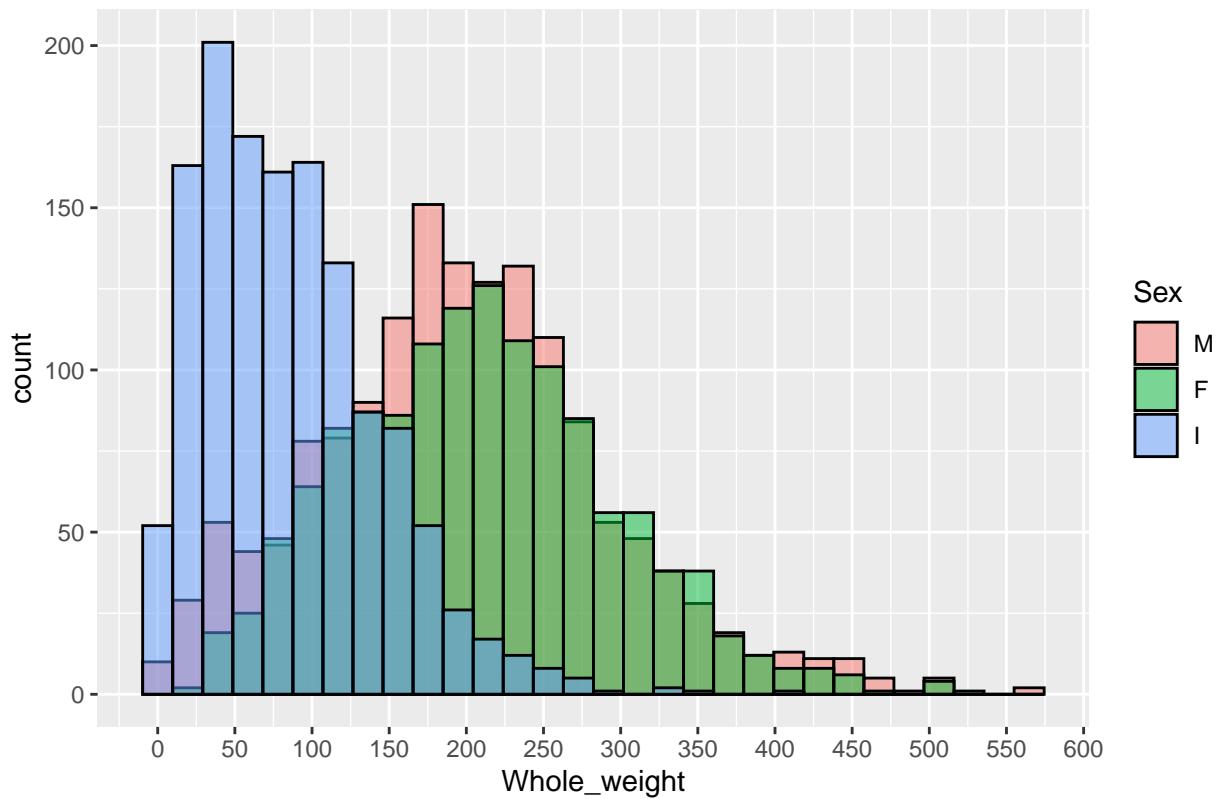


Para este par de variables se aprecia una mayor dispersión para los machos que para las hembras, pero no hay diferencias grandes entre sus distribuciones. Para los “infants” tenemos lo mismo que para las otras medidas de tamaño.

Pasamos a las medidas de peso, comenzando por el par “Sex-Whole weight”:

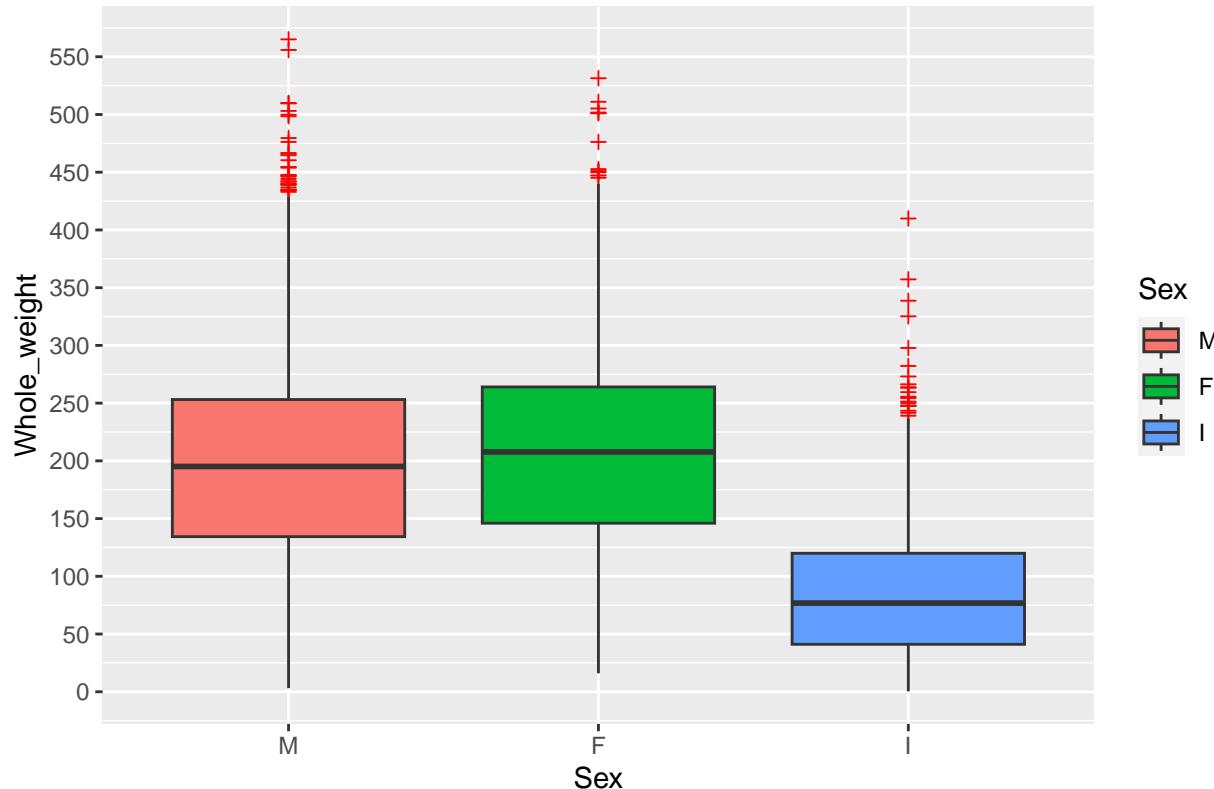
```
#sex-Whole Weight
ggplot(abalone, aes(x=Whole_weight, fill=Sex)) +
  geom_histogram(bins=30, color='black', alpha=0.5, position='identity')+
  labs(x='Whole_weight', title = 'Histograma de Whole_weight en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=400, by=50))+
```

Histograma de Whole_weight en función del sexo



```
ggplot(abalone, aes(y=Whole_weight, x=Sex)) +
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3) +
  labs(y='Whole_weight',
       title = 'Diagrama de cajas de la variable Whole_weight por sexo') +
  scale_y_continuous(breaks = seq(from=0, to=600, by=50))
```

Diagrama de cajas de la variable Whole_weight por sexo



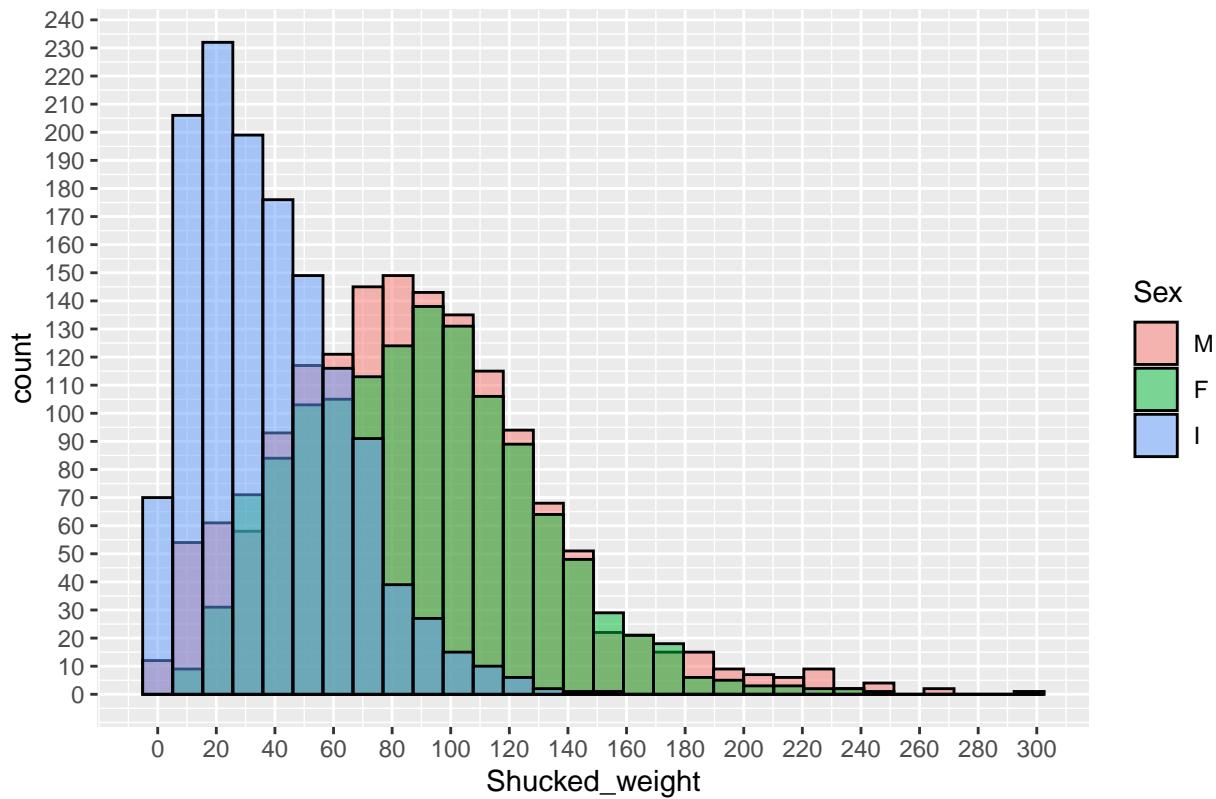
En este caso se puede apreciar que las distribución de “infant” está más separada de las otras dos debido a que es la más asimétrica, y ademárs, las variables de peso tenían un “skewness” positivo. Esto hace que la mayoría de los individuos de este grupo tome valores bajos para “Whole weight” y un solapamiento ligeramente menor.

Para el caso de “Sex-Shucked weight”:

```
#sex-Shucked_weight

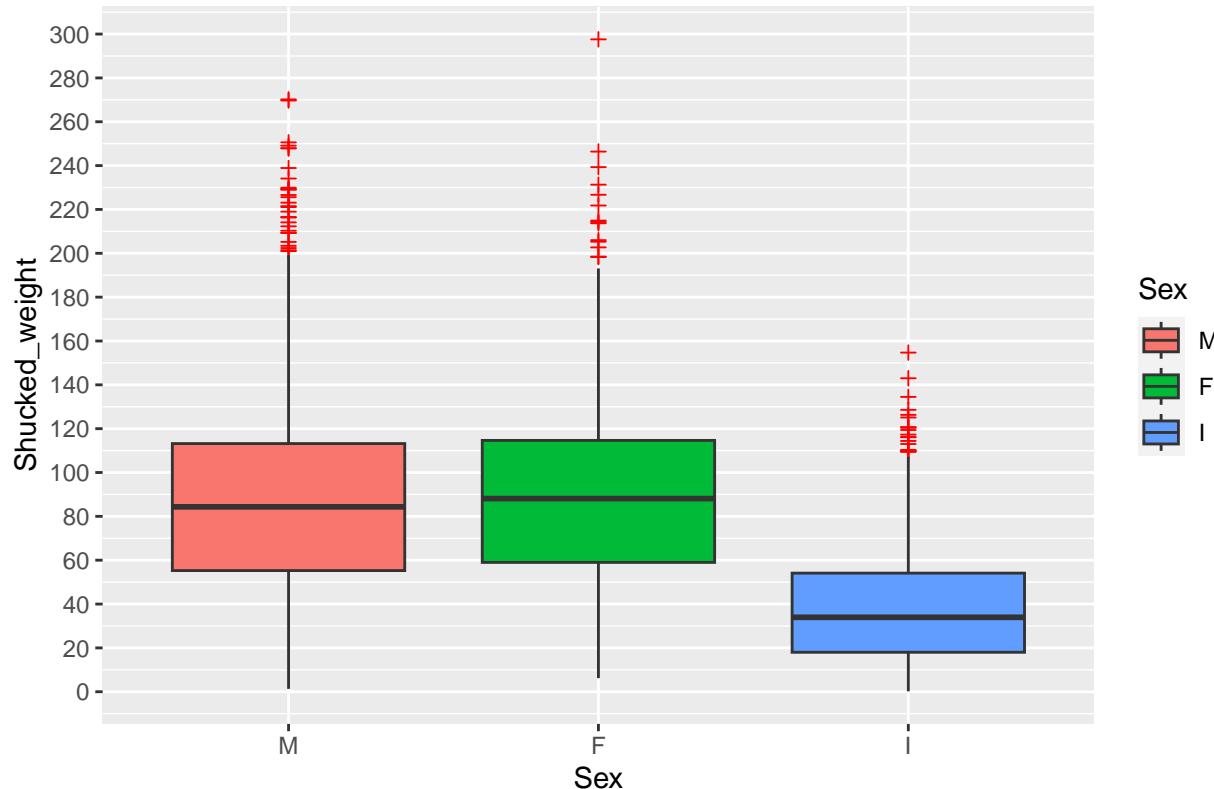
ggplot(abalone, aes(x=Shucked_weight, fill=Sex)) +
  geom_histogram(bins=30, color='black', alpha=0.5, position='identity')+
  labs(x='Shucked_weight', title = 'Histograma de Shucked_weight en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=250, by=10))+
```

Histograma de Shucked_weight en función del sexo



```
ggplot(abalone, aes(y=Shucked_weight, x=Sex)) +
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3) +
  labs(y='Shucked_weight',
       title = 'Diagrama de cajas de la variable Shucked_weight por sexo') +
  scale_y_continuous(breaks = seq(from=0, to=300, by=20))
```

Diagrama de cajas de la variable Shucked_weight por sexo



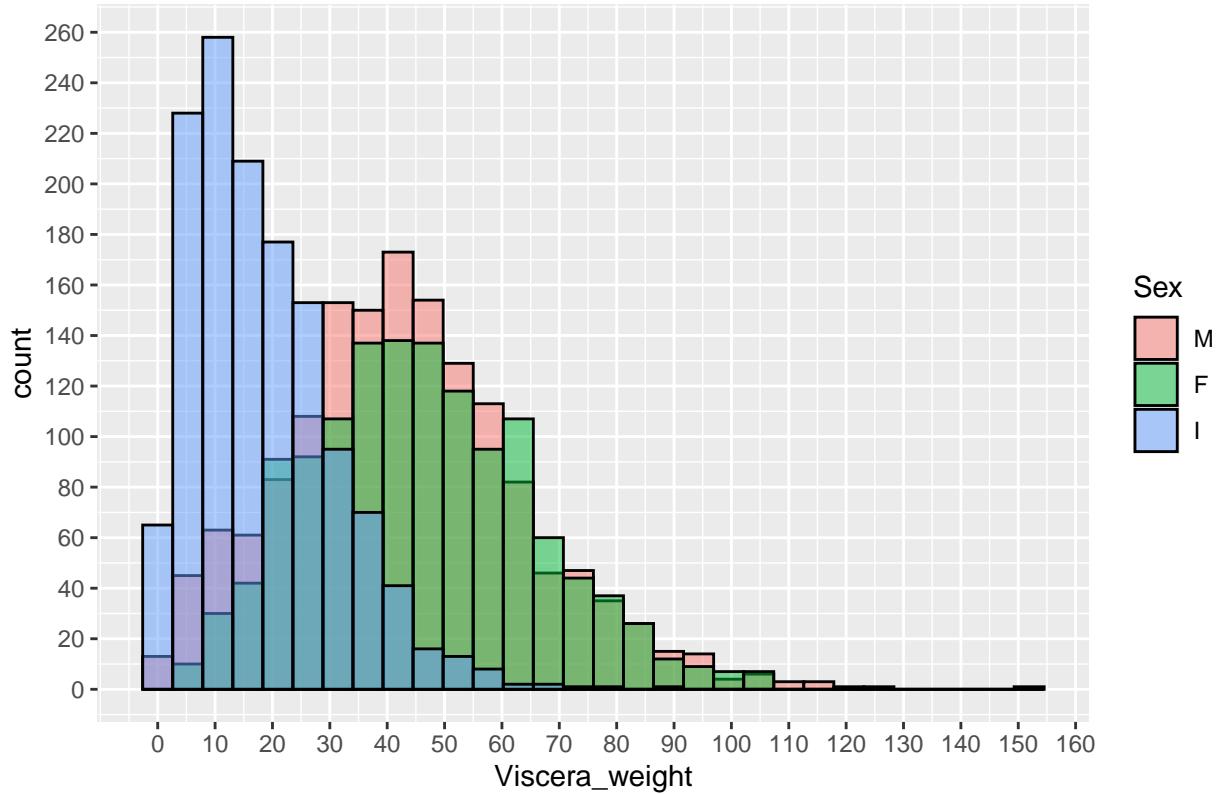
No hay tantas diferencias en los gráficos con respecto al par de variables anterior, igual puede ser que las distribuciones de “male” y “female” sean algo más asimétricas.

Para el par “Sex-Viscera weight”:

```
#sex-Viscera_weight

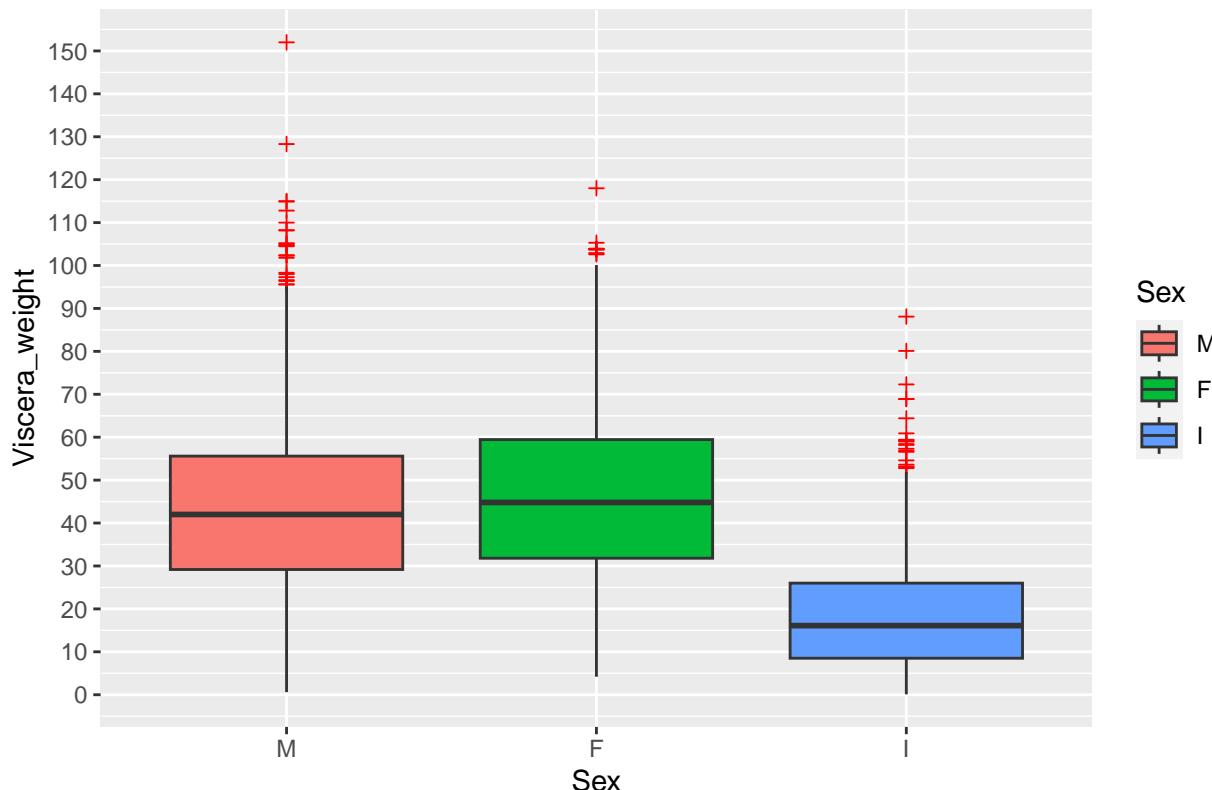
ggplot(abalone, aes(x=Viscera_weight, fill=Sex)) +
  geom_histogram(bins=30, color='black', alpha=0.5, position='identity')+
  labs(x='Viscera_weight', title = 'Histograma de Viscera_weight en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=260, by=20))+
```

Histograma de Viscera_weight en función del sexo



```
ggplot(abalone, aes(y=Viscera_weight, x=Sex)) +  
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Viscera_weight',  
       title = 'Diagrama de cajas de la variable Viscera_weight por sexo')+  
  scale_y_continuous(breaks = seq(from=0, to=170, by=10))
```

Diagrama de cajas de la variable Viscera_weight por sexo

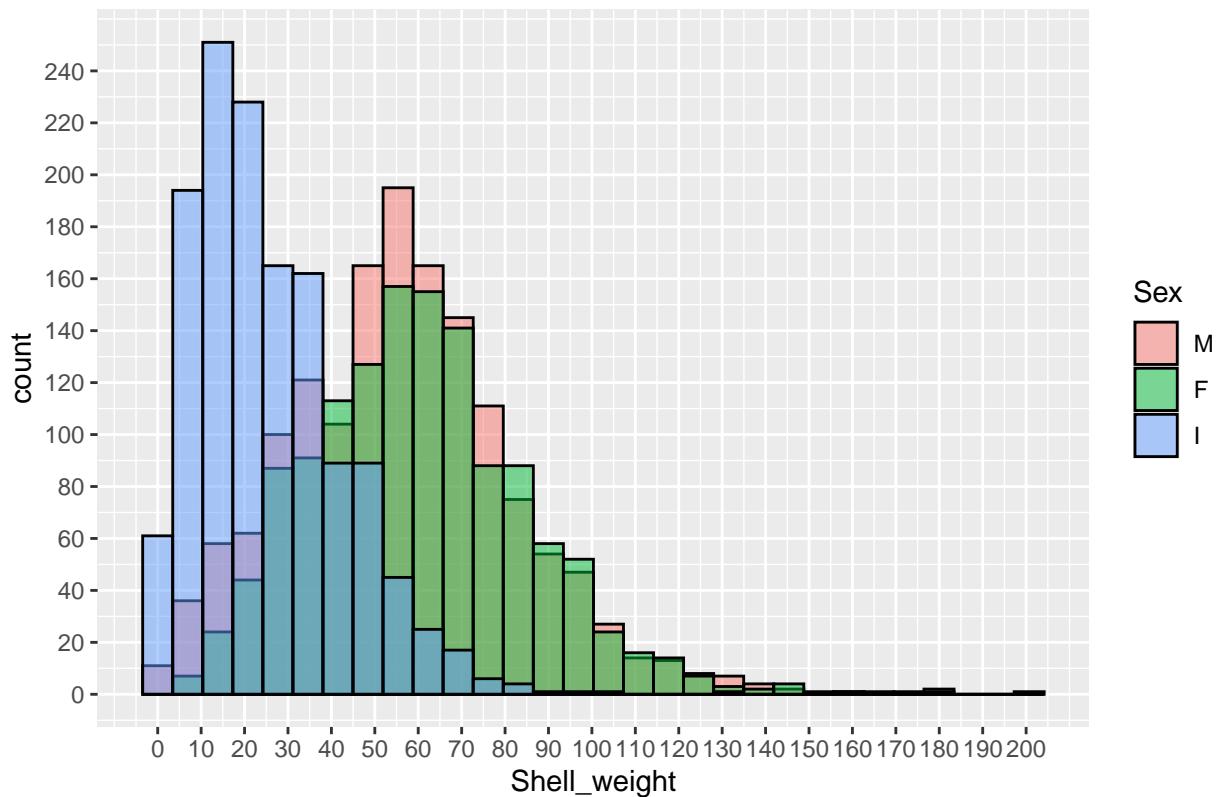


No hay diferencias apreciables, como mucho que las distribuciones son más simétricas y que la categoría “female” tiene muy pocos outliers para esta variable.

Ahora con el par “Sex-Shell weight”:

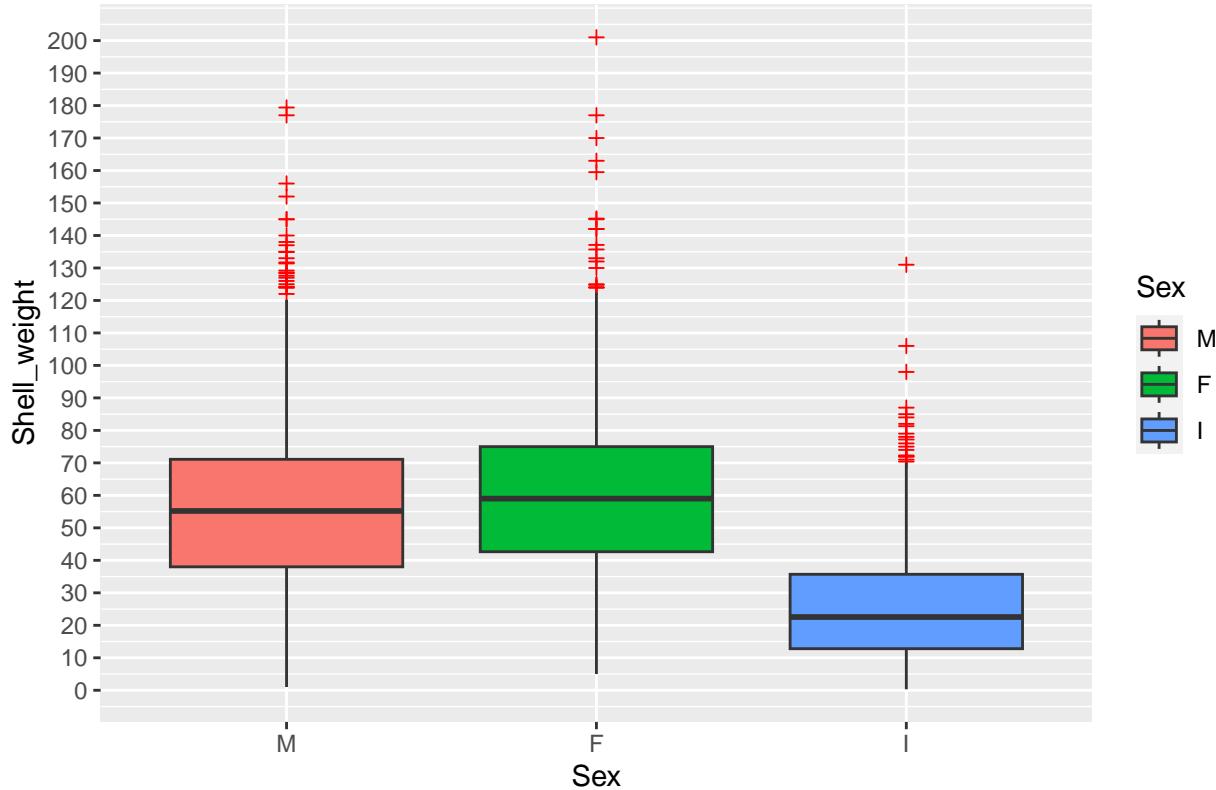
```
#sex-Shell_weight
ggplot(abalone, aes(x=Shell_weight, fill=Sex)) +
  geom_histogram(bins=30, color='black', alpha=0.5, position='identity')+
  labs(x='Shell_weight', title = 'Histograma de Shell_weight en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=250, by=20))+
```

Histograma de Shell_weight en función del sexo



```
ggplot(abalone, aes(y=Shell_weight, x=Sex)) +  
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Shell_weight',  
       title = 'Diagrama de cajas de la variable Shell_weight por sexo')+  
  scale_y_continuous(breaks = seq(from=0, to=200, by=10))
```

Diagrama de cajas de la variable Shell_weight por sexo



No hay mucho que añadir para esta variable que no se haya visto para las anteriores variables de peso, parece que estas variables quedan en su mayor parte explicadas por la variable del peso completo “whole weight”.

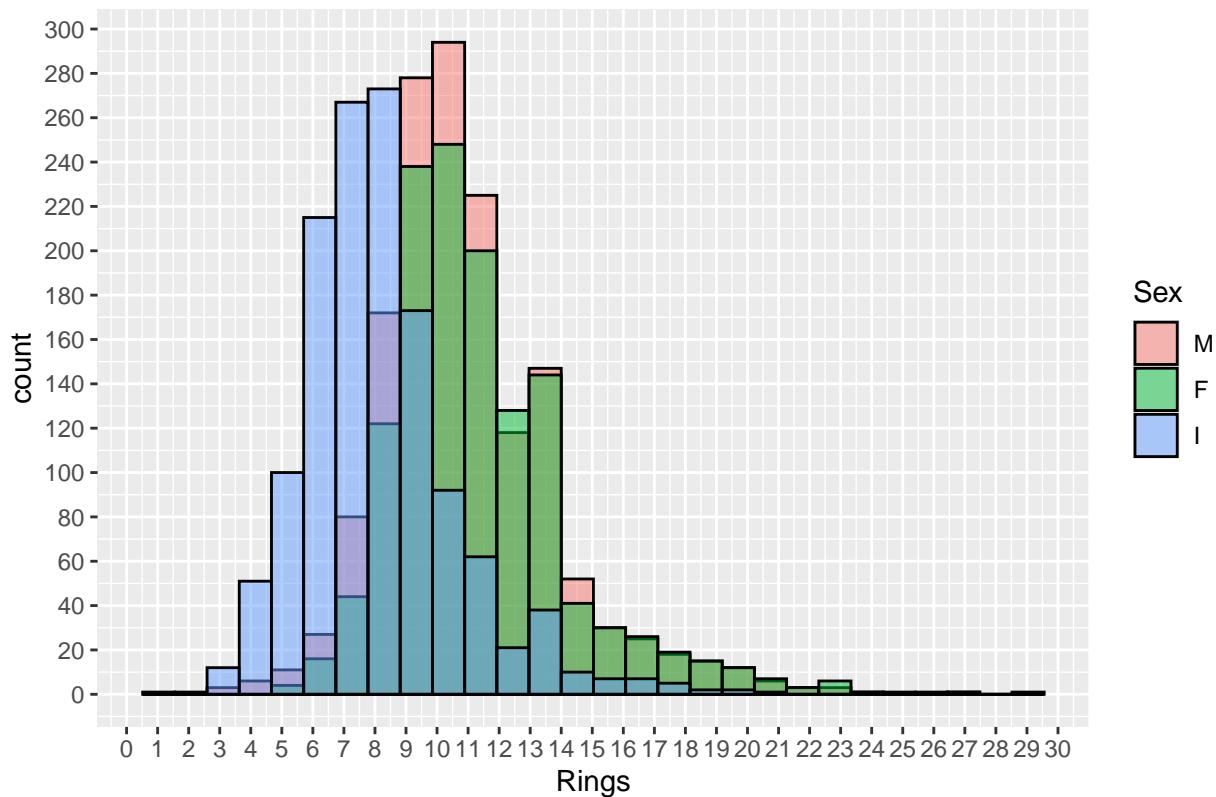
Con respecto a la variable de salida “Sex-Rings”:

```
#sex-Rings
```

```
ggplot(abalone, aes(x=Rings, fill=Sex)) +
  geom_histogram(bins=length(unique(abalone$Rings)), color='black', alpha=0.5,
                 position='identity')+
  labs(x='Rings', title = 'Histograma de Rings en función del sexo')+
  scale_y_continuous(breaks = seq(from=0, to=300, by=20))+
```

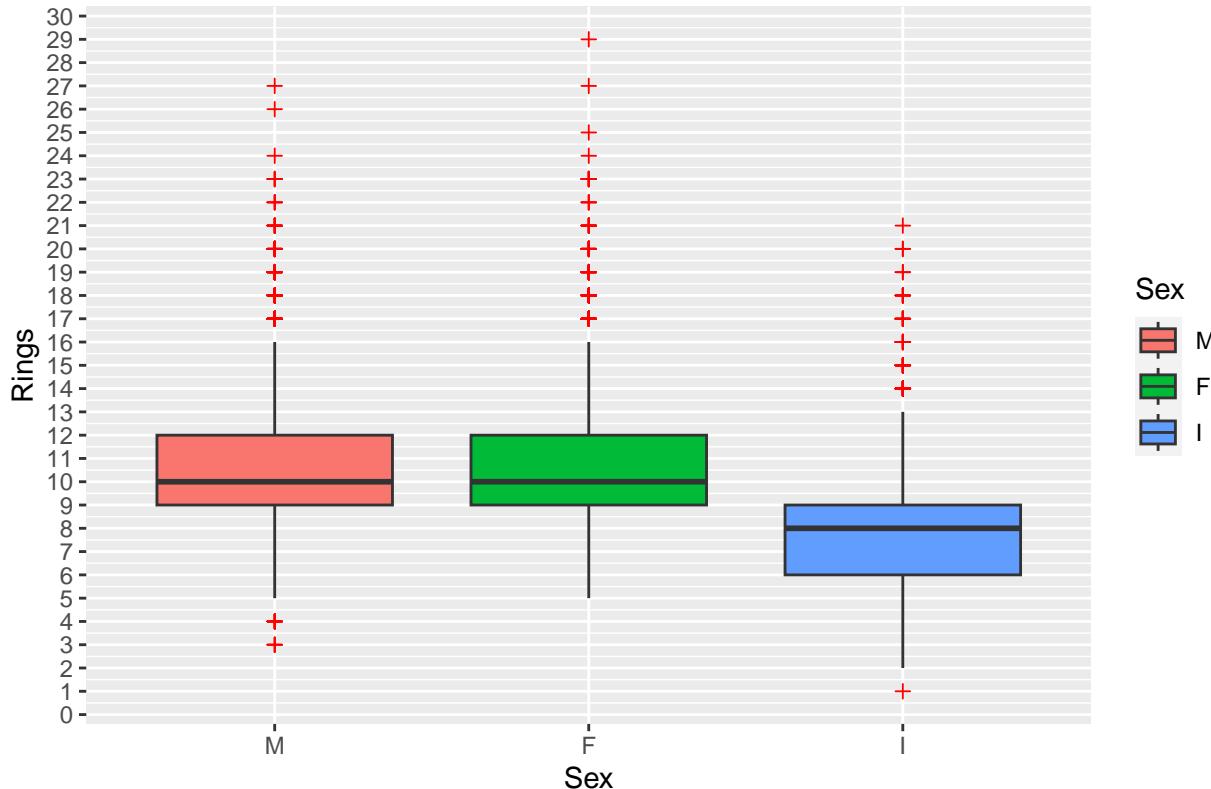
```
scale_x_continuous(breaks = seq(from=0, to=30, by=1))
```

Histograma de Rings en función del sexo



```
ggplot(abalone, aes(y=Rings, x=Sex)) +  
  geom_boxplot(aes(fill=Sex), outlier.colour = 'red', outlier.shape = 3)+  
  labs(y='Rings', title = 'Diagrama de cajas de la variable Rings por sexo')+  
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))
```

Diagrama de cajas de la variable Rings por sexo



En este caso es donde parece que hay el mayor solapamiento entre las categorías de la variable “Sex”, lo que es curioso por la relación entre el número de anillos y la edad del individuo. En la sección de irregularidades del apartado de procesamiento de los datos se plantean dos posibles hipótesis del motivo de esta ocurrencia.

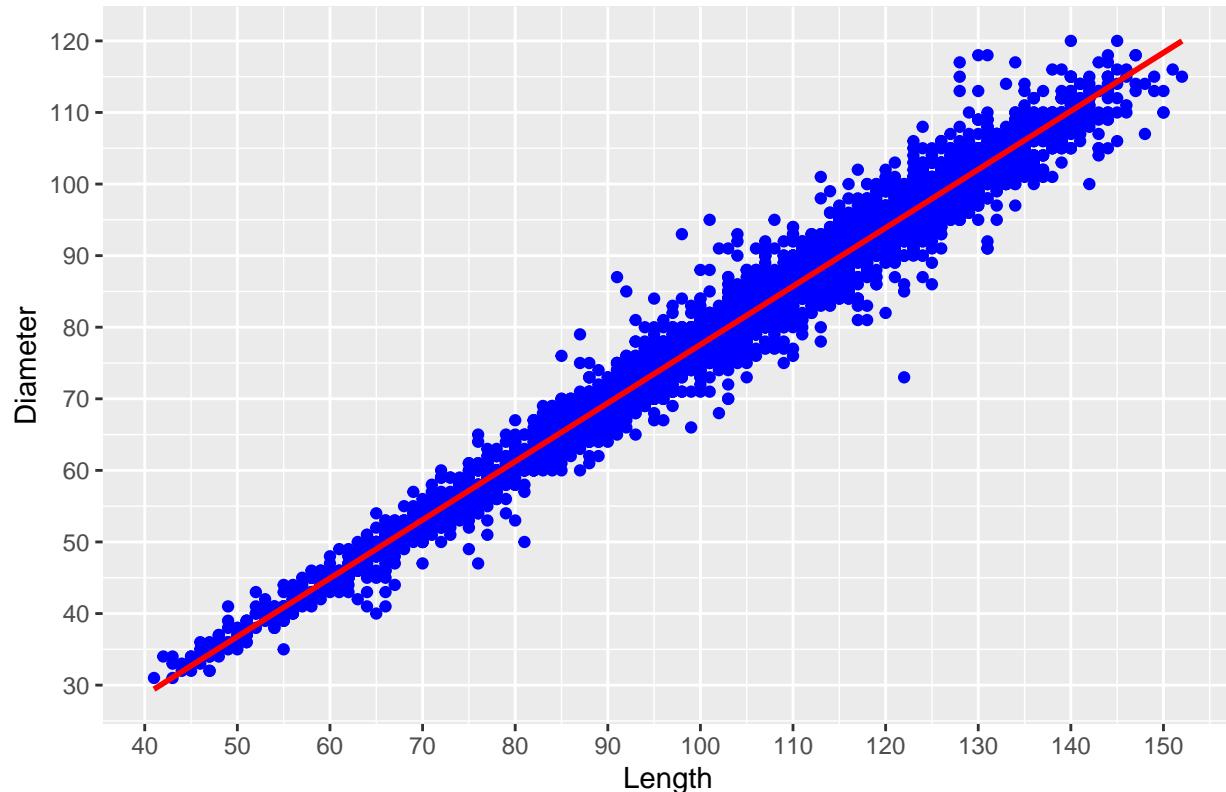
Una vez estudiados los pares con la variable categórica “Sex”, pasamos a estudiar los pares de variables numéricas empezando con las de tamaño. Este estudio es importante ya que permite ver outliers multivariantes, por lo tanto se empleará el conjunto de datos sin los outliers univariados a no ser que se especifique lo contrario. También es importante porque permite visualizar las interacciones y la relación que hay entre las variables. Para representar los pares de variables se usarán gráficos de dispersión y Se seguirá el mismo código de colores descrito anteriormente.

Par de variables “Length-Diameter”:

```
# Length-Diameter
ggplot(abalone[-abalone.outliers,],aes(x=Length, y=Diameter)) + geom_point(color='Blue')+geom_smooth(method=lm, color='Red')+
  labs(title = 'Diagrama de dispersión entre Length y Diameter')+
  scale_y_continuous(breaks = seq(from=0, to=150, by=10))+scale_x_continuous(breaks = seq(from=0, to=160, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Length y Diameter



Podemos distinguir un punto que puede ser un posible dato anómalo.

```
abalone[-abalone.outliers,] %>% filter(Diameter<75, Length>120)
```

```
##   Sex Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## 1   M     122        73      31       215.3         97.6          49.8
##   Shell_weight Rings
## 1           54      9
```

Este tipo de datos atípicos pueden afectar a la calidad de la recta de regresión, aunque su influencia no es tan grande debido a la gran cantidad de observaciones que siguen la distribución. Recordamos que la media es sensible a datos de valores extremos, por lo que lo más indicado para la regresión lineal sería eliminar los datos atípicos.

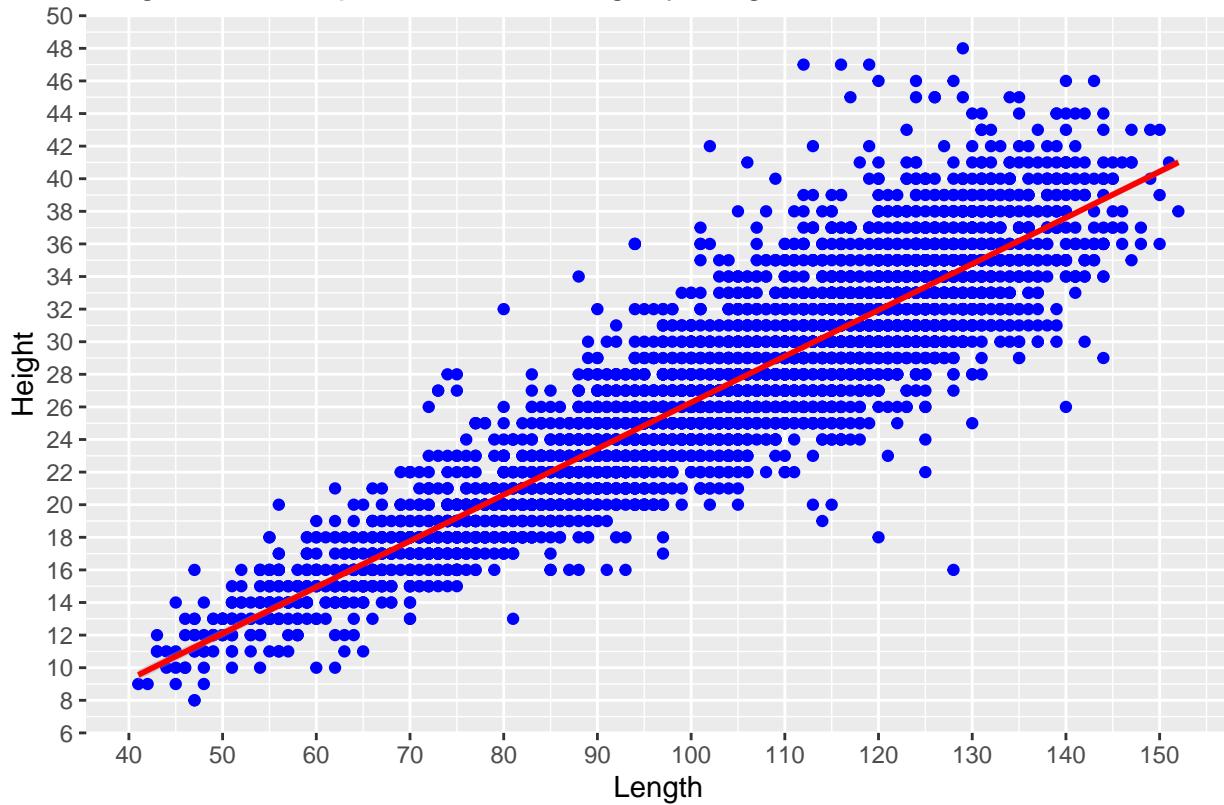
Por otro lado, se puede apreciar una correlación bastante alta, y por lo general parece que la forma ovalada de la concha tiene a mantenerse en las mismas proporciones conforme esta crece. Para este par de variables parece que hay homocedasticidad, ya que tenemos un número mayor de individuos de tamaño medio-grande y eso puede afectar a la varianza.

Para el par de variables "Length-Height":

```
ggplot(abalone[-abalone.outliers,], aes(x=Length, y=Height)) + geom_point(color='Blue')+
  geom_smooth(method=lm, color='Red')+
  labs(title = 'Diagrama de dispersión entre Length y Height')+
  scale_y_continuous(breaks = seq(from=0, to=50, by=2))+ 
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Length y Height

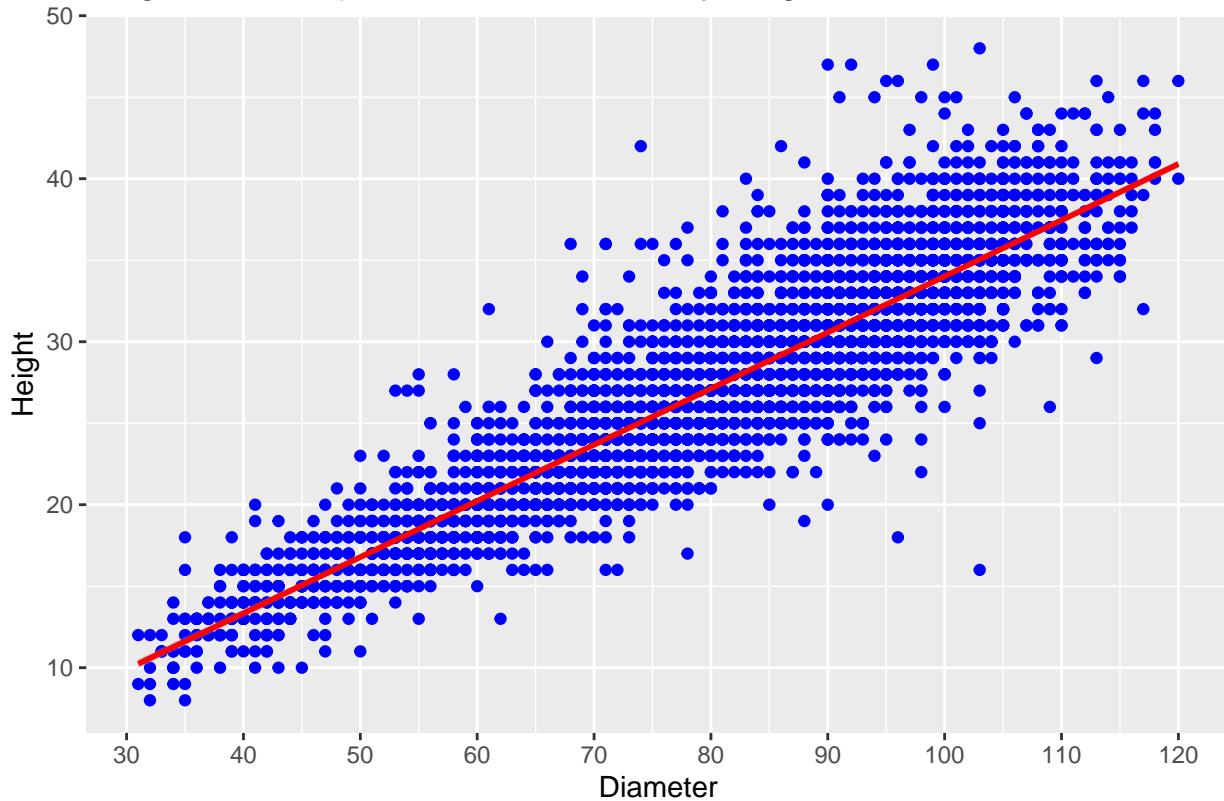


En este caso aunque también haya bastante correlación, la dispersión de los datos es mayor. Esto es debido a que la variable “Height” no solo comprende parte de la concha, si no parte de la vianda también; lo que puede producir esa mayor variabilidad.

Si la representamos con respecto a “Diameter”:

```
ggplot(abalone[-abalone.outliers,],aes(x=Diameter, y=Height)) + geom_point(color='Blue')+  
  geom_smooth(method=lm, color='Red') +  
  labs(title = 'Diagrama de dispersión entre Diameter y Height') +  
  scale_y_continuous(breaks = seq(from=0, to=150, by=10)) +  
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))  
  
## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Diameter y Height



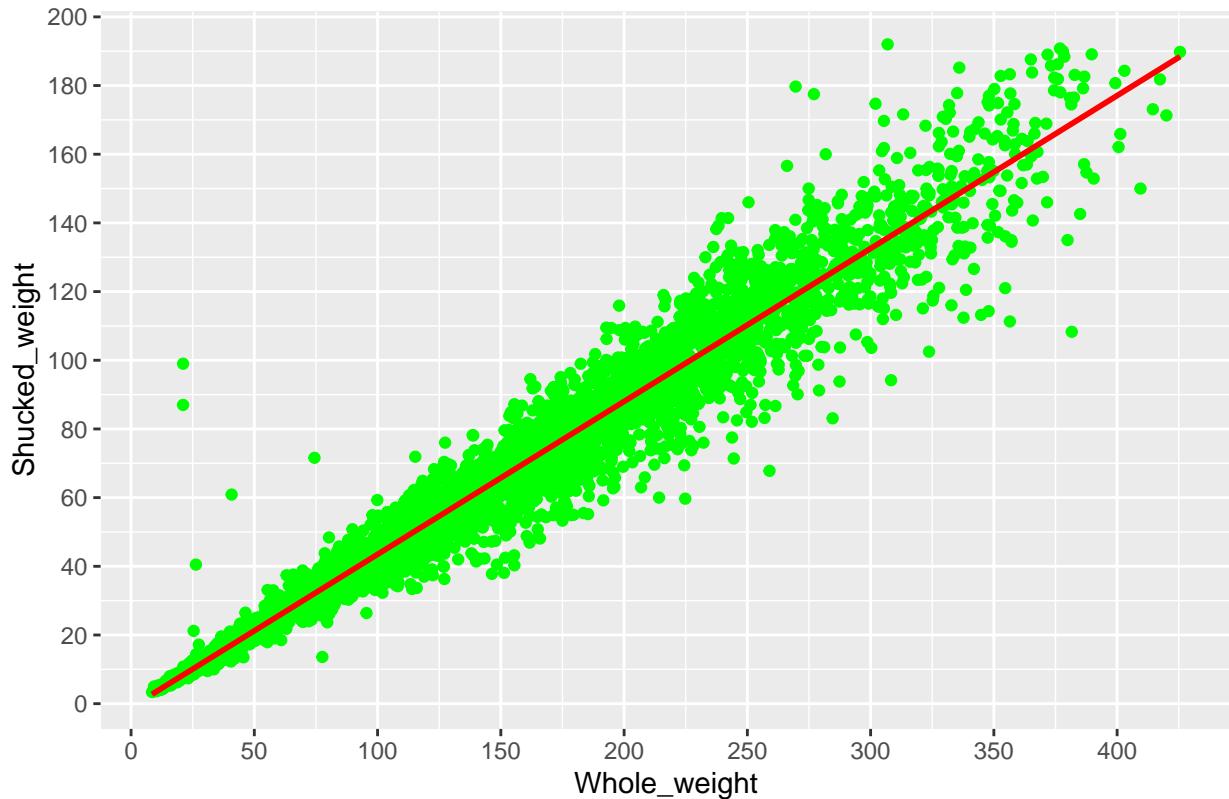
Se obtiene un resultado muy similar al gráfico anterior, con mucha dispersión en la variable height. Cabe destacar que la heterocedasticidad puede ser un problema a la hora de modelar la regresión entre las dos variables de tamaño y la variable "Height".

Ahora se estudiarán las variables de peso por pares. Comenzando por "Whole weight-Shucked weight"

```
# Whole_weight-Shucked_weight
ggplot(abalone[-abalone.outliers,],aes(x=Whole_weight, y=Shucked_weight)) +
  geom_point(color='Green')+ geom_smooth(method=lm, color='Red')+
  labs(title = 'Diagrama de dispersión entre Whole_weight y Shucked_weight')+
  scale_y_continuous(breaks = seq(from=0, to=300, by=20))+ 
  scale_x_continuous(breaks = seq(from=0, to=600, by=50))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Whole_weight y Shucked_weight



Como podemos comprobar, aunque haya una correlación alta entre ambas variables, la dispersión de los puntos aumenta conforme aumenta el valor de las variables, incluso sin considerar los outliers univariados, por lo que se podría concluir que tenemos cierta heterocedasticidad. También hay unos cuantos outliers que podrían afectarnos a la regresión si consideramos la interacción de estas dos variables. Estos puntos son:

```
abalone[-abalone.outliers,] %>% filter(Whole_weight<75,Shucked_weight>40)
```

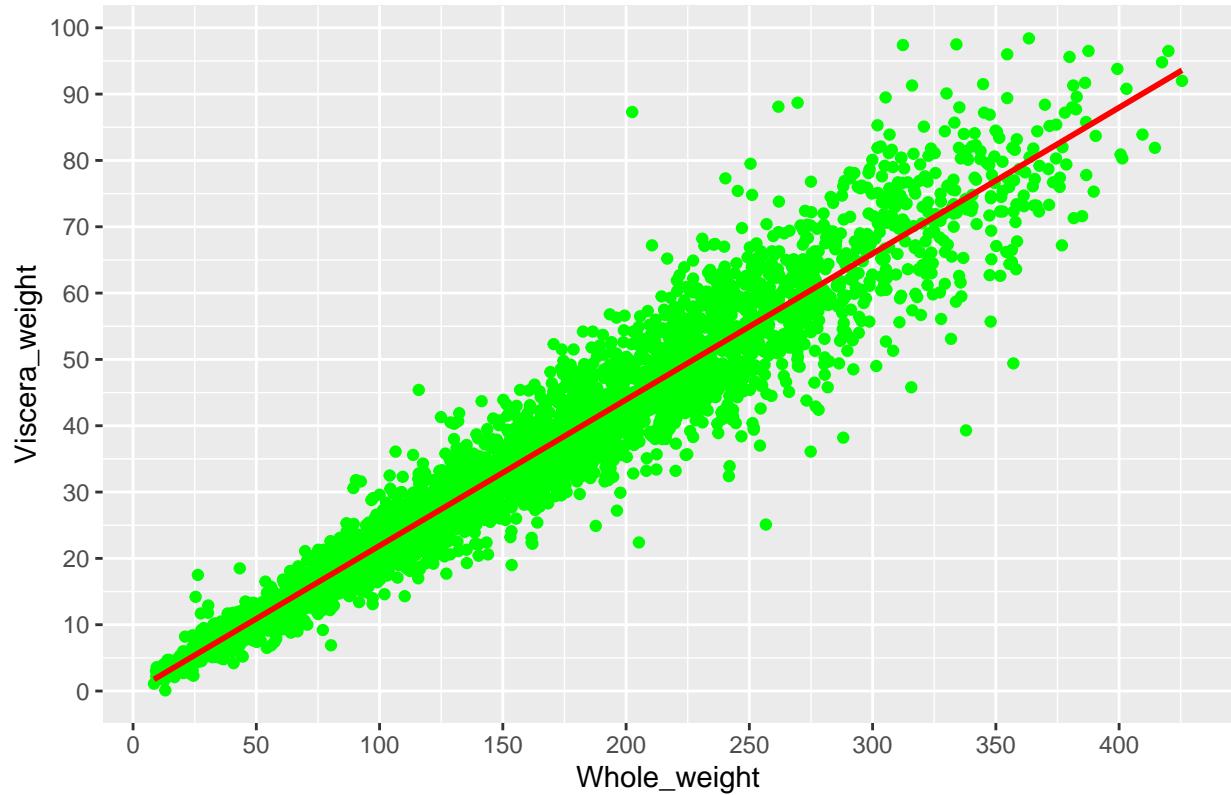
```
##   Sex Length Diameter Height Whole_weight Shucked_weight Viscera_weight
## 1   I      71       54     15       40.8       60.9          9.2
## 2   I      91       66     20       74.4       71.6         15.5
## 3   I      55       41     14       21.1       99.0          3.8
## 4   I      62       45     14       21.1       87.0          3.0
## 5   I      95       73     20       26.3       40.5         17.5
##   Shell_weight Rings
## 1        11.9     7
## 2        22.0     8
## 3        6.3      5
## 4        8.0      5
## 5        24.6     7
```

Para el par de variables “Whole weight-Viscera weight”:

```
ggplot(abalone[-abalone.outliers,],aes(x=Whole_weight, y=Viscera_weight)) +
  geom_point(color='Green')+  geom_smooth(method=lm, color='Red')+
  labs(title = 'Diagrama de dispersión entre Whole_weight y Viscera_weight')+
  scale_y_continuous(breaks = seq(from=0, to=150, by=10))+
  scale_x_continuous(breaks = seq(from=0, to=600, by=50))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Whole_weight y Viscera_weight



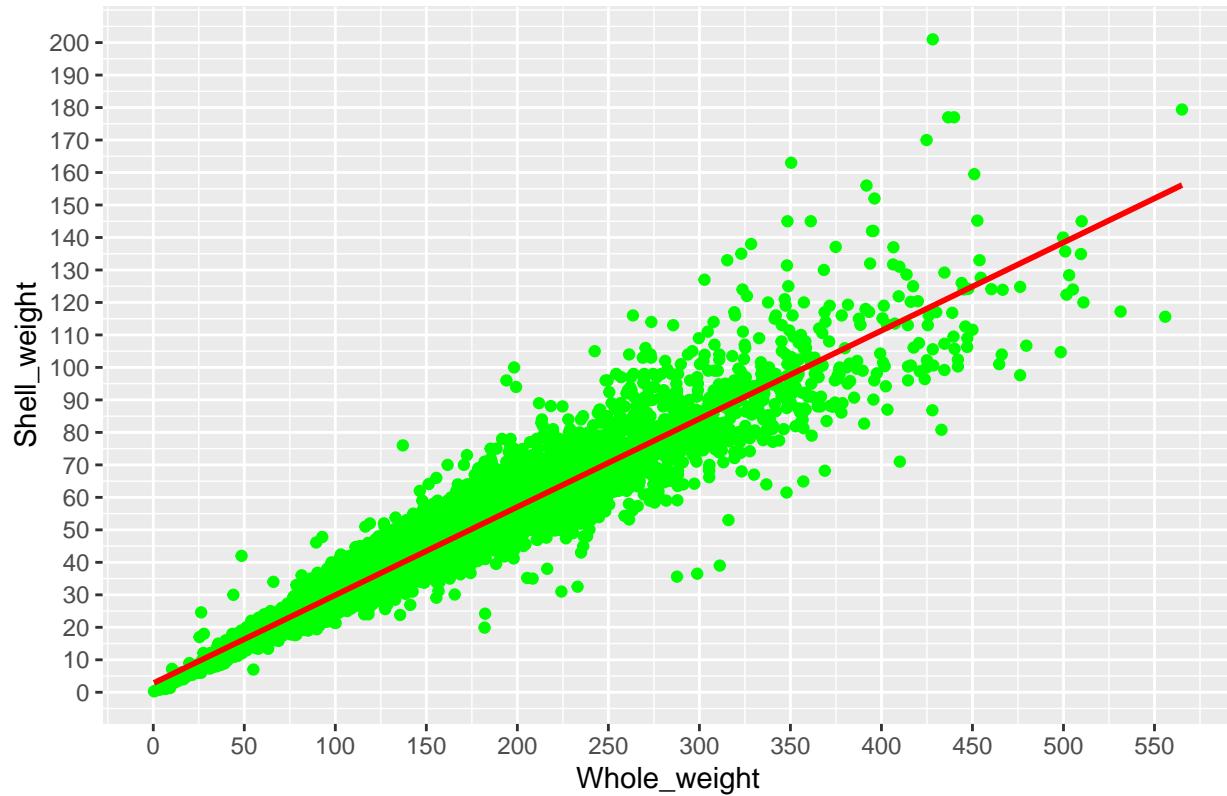
Incluso sin considerar los outliers univariados, también tenemos bastante heterocedasticidad, lo cual es un problema a la hora de modelar una regresión. También se podría considerar que hay outliers multivariantes, aunque no es tan claro ya que podría ser por la dispersión que hay en los datos.

Comprobamos si tenemos el mismo problema en el par “Whole weight-Shell weight”:

```
# Whole_weight-Shell_weight
ggplot(abalone[-abalone.outliers],aes(x=Whole_weight, y=Shell_weight)) +
  geom_point(color='Green')+  geom_smooth(method=lm, color='Red')+
  labs(title = 'Diagrama de dispersión entre Whole_weight y Shell_weight')+
  scale_y_continuous(breaks = seq(from=0, to=200, by=10))+
  scale_x_continuous(breaks = seq(from=0, to=600, by=50))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Whole_weight y Shell_weight



También hay mucha dispersión en los datos, de hecho en este caso la recta de regresión dibujada sobre el diagrama de dispersión se ve desplazada y no modela bien los primeros datos de valores bajos.

Esto nos lleva a cuestionarnos si el problema está en la variable “Whole weight”, para ello estudiamos los diagramas de dispersión del resto de variables de peso.

Con el par “Shucked weight-Viscera weight”:

```
# Shucked_weight-Viscera_Weight
ggplot(abalone[-abalone.outliers,], aes(x=Shucked_weight, y=Viscera_weight)) +
  geom_point(color='Green') + geom_smooth(method=lm, color='Red') +
  labs(title = 'Diagrama de dispersión entre Shucked_weight y Viscera_weight') +
  scale_y_continuous(breaks = seq(from=0, to=150, by=10)) +
  scale_x_continuous(breaks = seq(from=0, to=300, by=20))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Shucked_weight y Viscera_weight



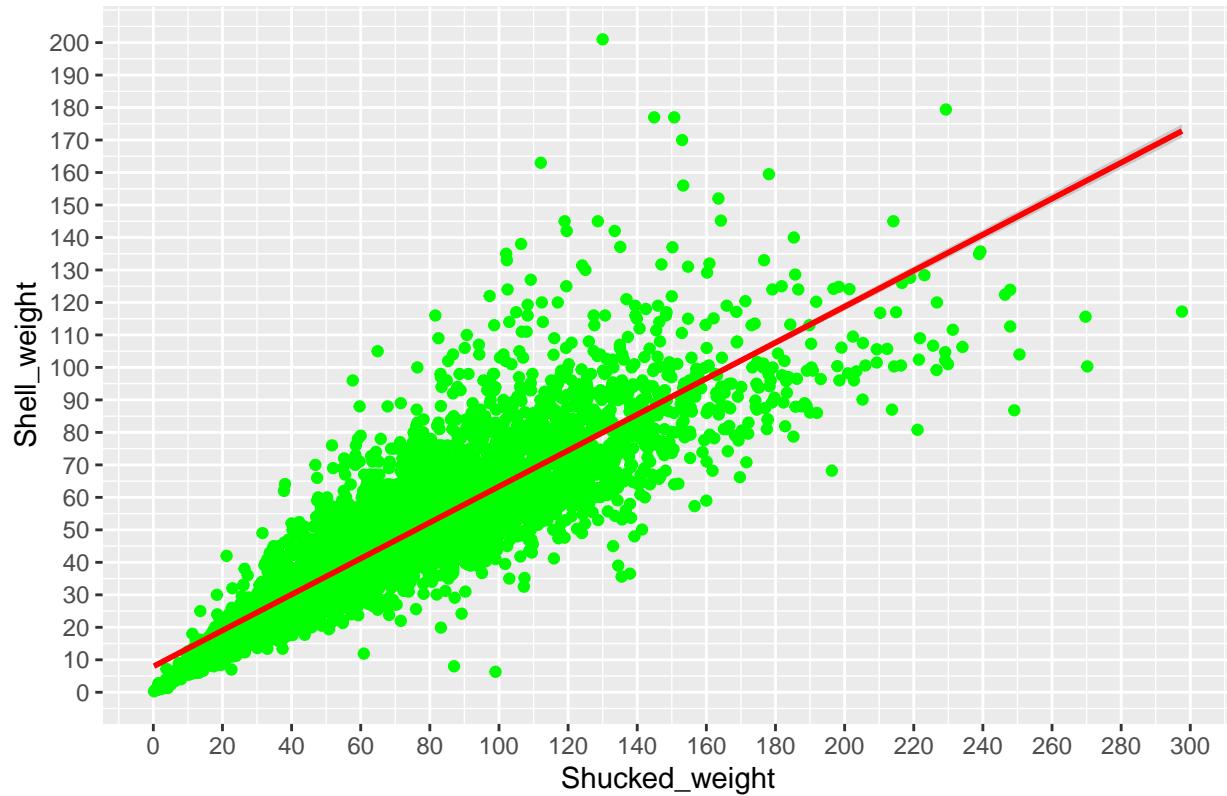
Como podemos observar hay mayor dispersión, por lo que podemos descartar que la causa del problema sea la variable “Whole weight”.

Estudiando los dos últimos pares de variables de peso, “Shucked weight-Shell weight” y “Viscera weight-Shell weight”:

```
# Shucked_weight-Shell_weight
ggplot(abalone,aes(x=Shucked_weight, y=Shell_weight)) + geom_point(color='Green')+ geom_smooth(method=...
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

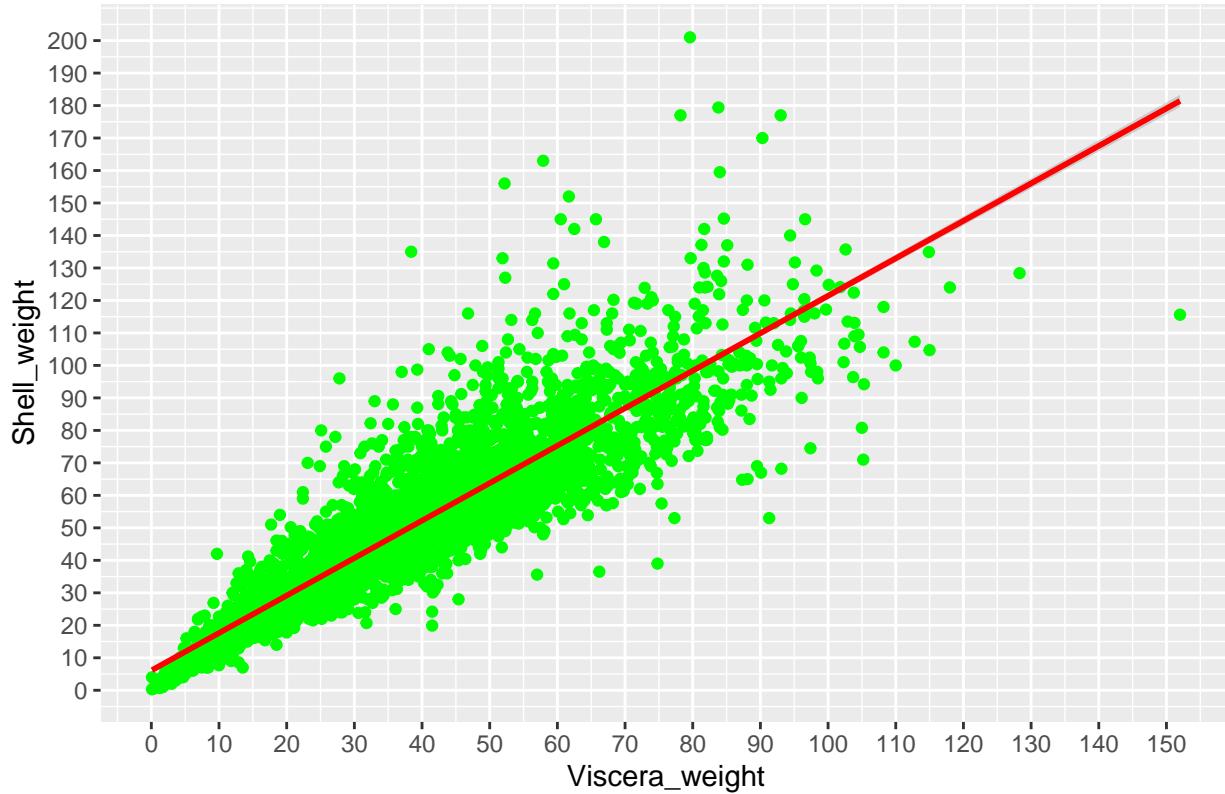
Diagrama de dispersión entre Shucked_weight y Shell_weight



```
# Viscera_weight-Shell_weight
ggplot(abalone,aes(x=Viscera_weight, y=Shell_weight)) + geom_point(color='Green')+  geom_smooth(method=...
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Viscera_weight y Shell_weight



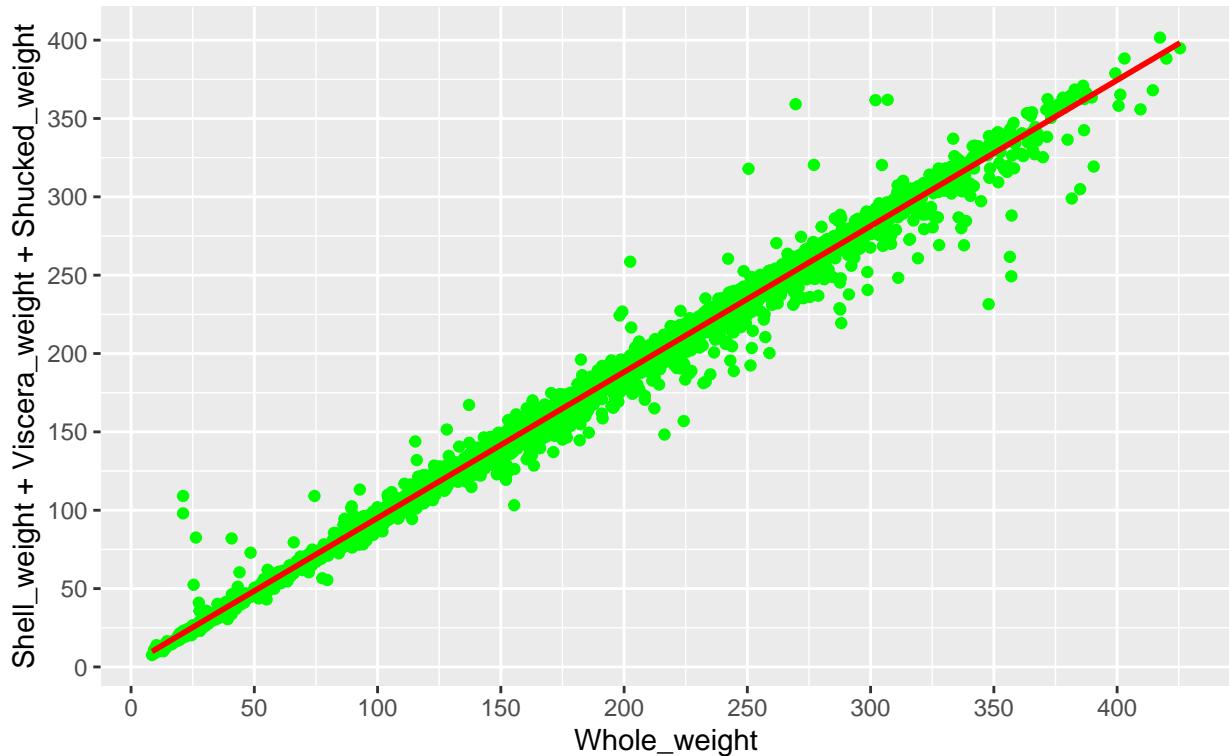
Tampoco tenemos resultados favorables respecto a la dispersión de los datos. Esto sería un problema de cara a la regresión si utilizamos términos de interacción entre las variables de peso, ya que la dispersión van a depender de ellas y una de las asunciones para la regresión es que haya homocedasticidad.

En un principio pensaríamos en aplicar algún tipo de transformación de las variables; pero por la descripción de las variables, sabemos que "Whole weight" es el peso total del individuo, y que el resto de las variables de peso corresponden a pesos de las distintas partes (concha, vísceras y vianda). ¿Y si probásemos comparar el peso total con la suma del peso de las distintas partes? Obtendríamos el siguiente gráfico de dispersión:

```
# Si sumamos las de peso:
ggplot(abalone [-abalone.outliers],aes(x=Whole_weight, y=Shell_weight+Viscera_weight
+Shucked_weight)) +
geom_point(color='Green')+ geom_smooth(method=lm, color='Red')+
labs(title = 'Diagrama de dispersión entre Whole_weight y la suma del resto de las
variables de peso')+
scale_y_continuous(breaks = seq(from=0, to=400, by=50))+
scale_x_continuous(breaks = seq(from=0, to=500, by=50))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Whole_weight y la suma del resto de las variables de peso



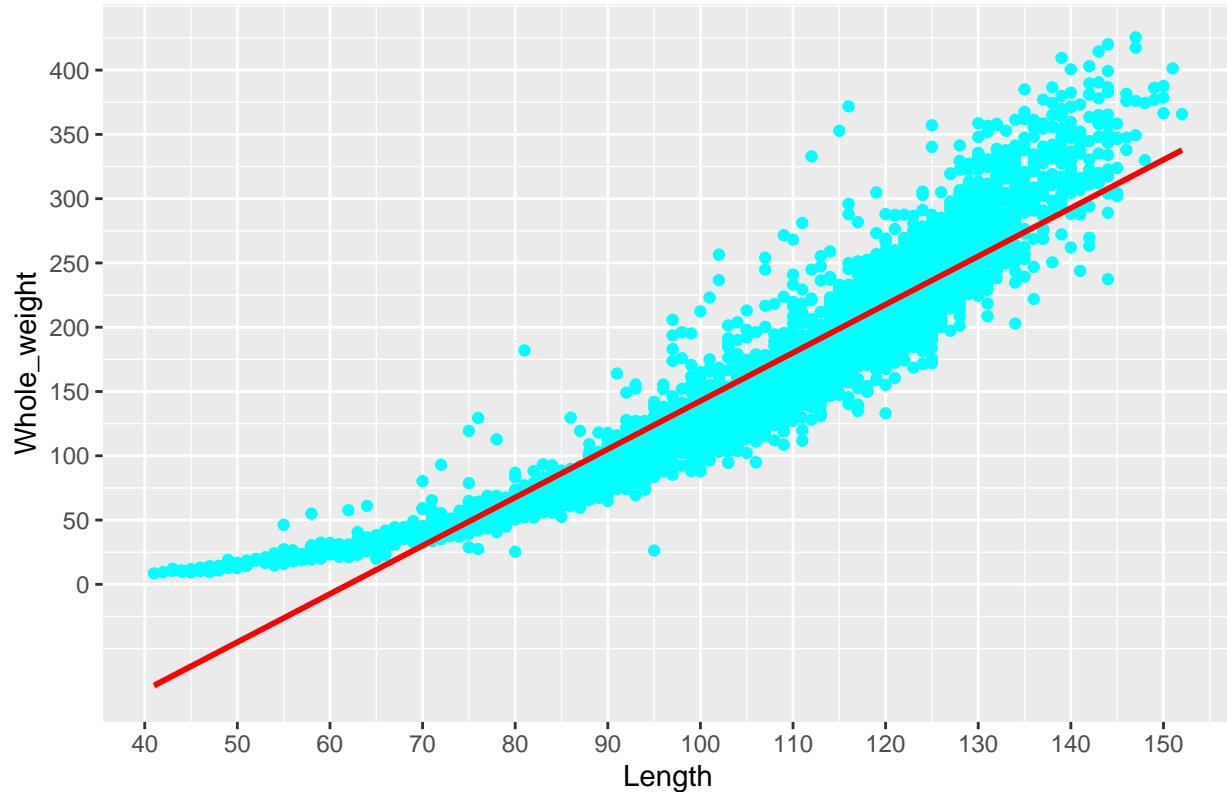
Se obtiene un mejor resultado en términos de homocedasticidad. El problema que teníamos es que para cada individuo la proporción de pesos de sus distintas partes podía variar en gran medida, esto se hace más aparente conforme el individuo crece. Por eso si sólo se comparan dos variables de peso puede haber mucha dispersión en los datos, especialmente para los individuos de mayor peso. Cabe destacar que todavía existe cierta dispersión, pero en este caso parece que es más uniforme conforme recorremos la recta de regresión. Esta dispersión se puede explicar en parte, si volvemos a leer la descripción de las variables, al peso perdido tras el sangrado y el secado en las variables "Viscera weight" y "Shell weight".

Ahora sería interesante estudiar la relación entre las variables de tamaño y de peso, como hemos visto la variable "Whole weight" está muy relacionada con la suma del resto de variables de peso, por lo que se utilizará esta principalmente. Empezando por "Length-Whole weight" tenemos:

```
ggplot(abalone [-abalone.outliers,],aes(x=Length, y=Whole_weight)) + geom_point(color='Cyan')+ geom_smooth()
  labs(title = 'Diagrama de dispersión entre Length y Whole_weight')+
  scale_y_continuous(breaks = seq(from=0, to=400, by=50))+ 
  scale_x_continuous(breaks = seq(from=0, to=150, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Length y Whole_weight

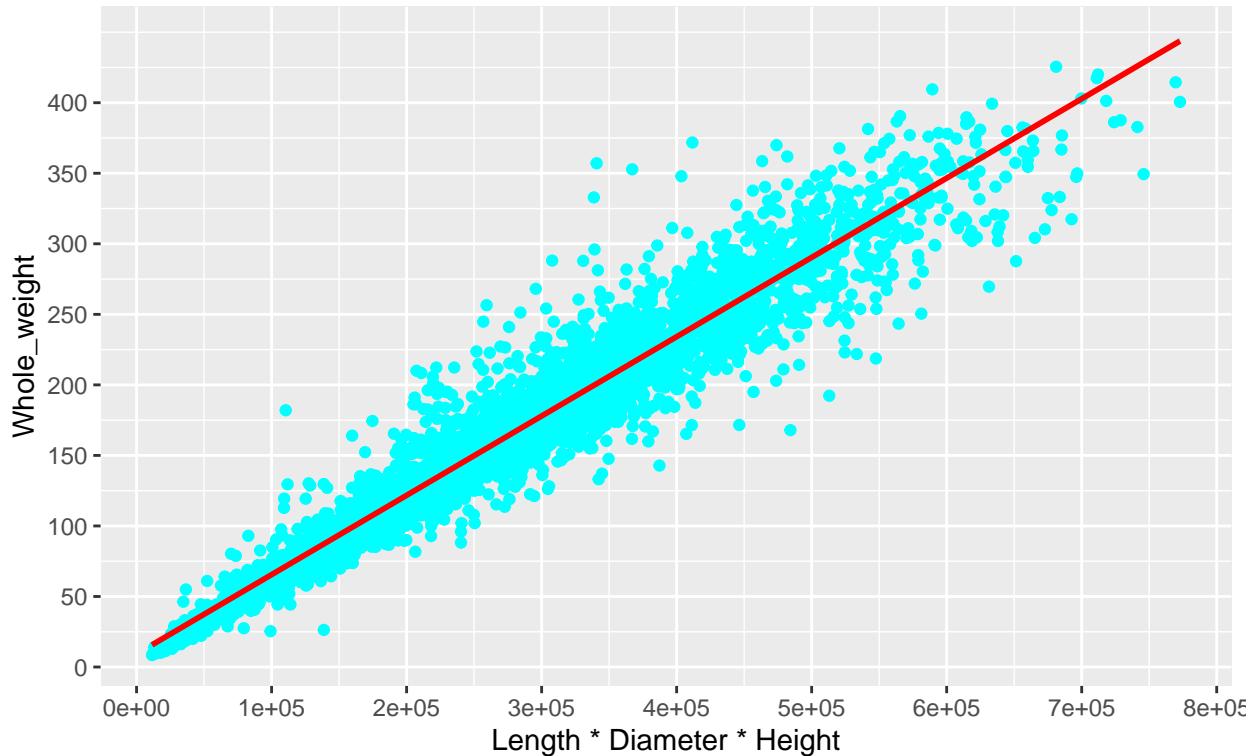


Podemos apreciar que el crecimiento del peso no se describe de forma lineal, si no más bien de forma cuadrática o cúbica. Esto es lógico ya que si asumimos una densidad constante, conforme el individuo crece en todas las direcciones del espacio, es decir que su volumen aumenta, su peso también aumentará. Si representamos el peso total con respecto al producto de las medidas de tamaño, tendremos:

```
ggplot(abalone [-abalone.outliers,], aes(x=Length*Diameter*Height, y=Whole_weight)) +
  geom_point(color='Cyan')+  geom_smooth(method=lm, color='Red')+
  labs(title = 'Diagrama de dispersión entre el producto de las variables de tamaño
y Whole_weight')+
  scale_y_continuous(breaks = seq(from=0, to=400, by=50))+
  scale_x_continuous(breaks = seq(from=0, to=800000, by=100000))

## `geom_smooth()` using formula = 'y ~ x'
```

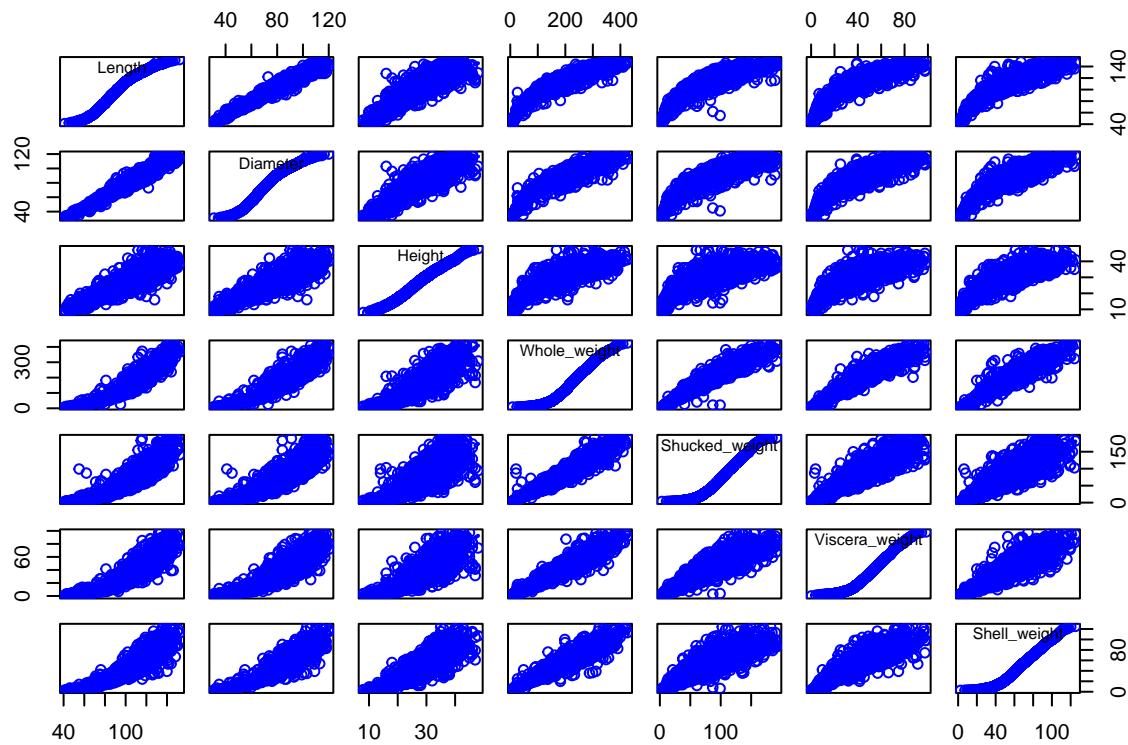
Diagrama de dispersión entre el producto de las variables de tamaño y Whole_weight



De este gráfico podemos concluir que el peso tiene una alta correlación con el producto de las tres variables de tamaño, aunque para obtener la relación exacta, habría que tener en cuenta las ecuaciones que describen la forma del abulón.

A continuación se muestra una tabla resumen con los diagramas de dispersión para cada par de variables numéricas, y en la diagonal central el “QQ plot” de cada variable.

```
# Resumen
scatterplotMatrix(abalone[-abalone.outliers,] %>% select(-Sex,-Rings), diagonal = list(method="qqplot"))
```



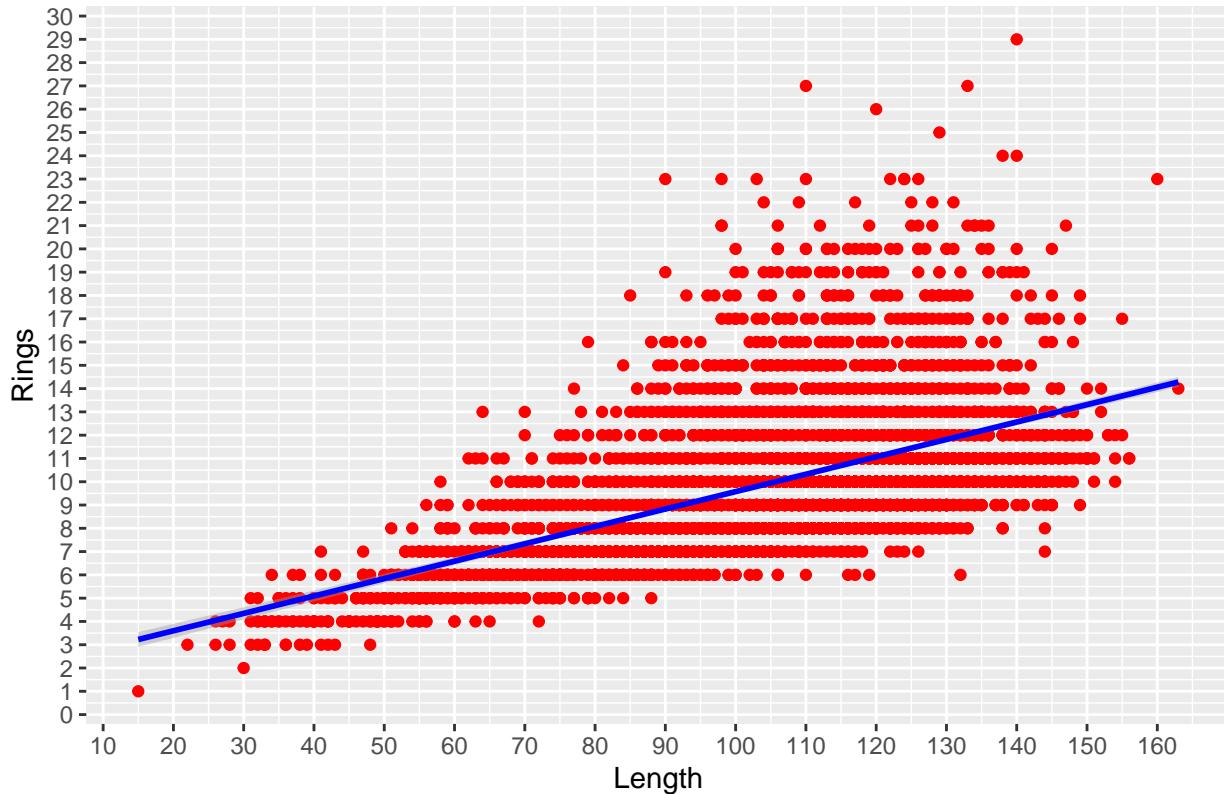
Gráficas bivariadas entre los atributos y la variable de salida

Una vez analizadas las interacciones de las variables de entrada entre ellas, procedemos a analizar las variables numéricas continuas con respecto a la de salida. Para este caso se han incluido los outliers, ya que se podría considerar necesario modelar los valores extremos de anillos. Empezando por “Length-Rings”:

```
# Length-Rings
ggplot(abalone,aes(x=Length, y=Rings)) + geom_point(color='Red')+ geom_smooth(method=lm, color='Blue')
  labs(title = 'Diagrama de dispersión entre Length y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Length y Rings



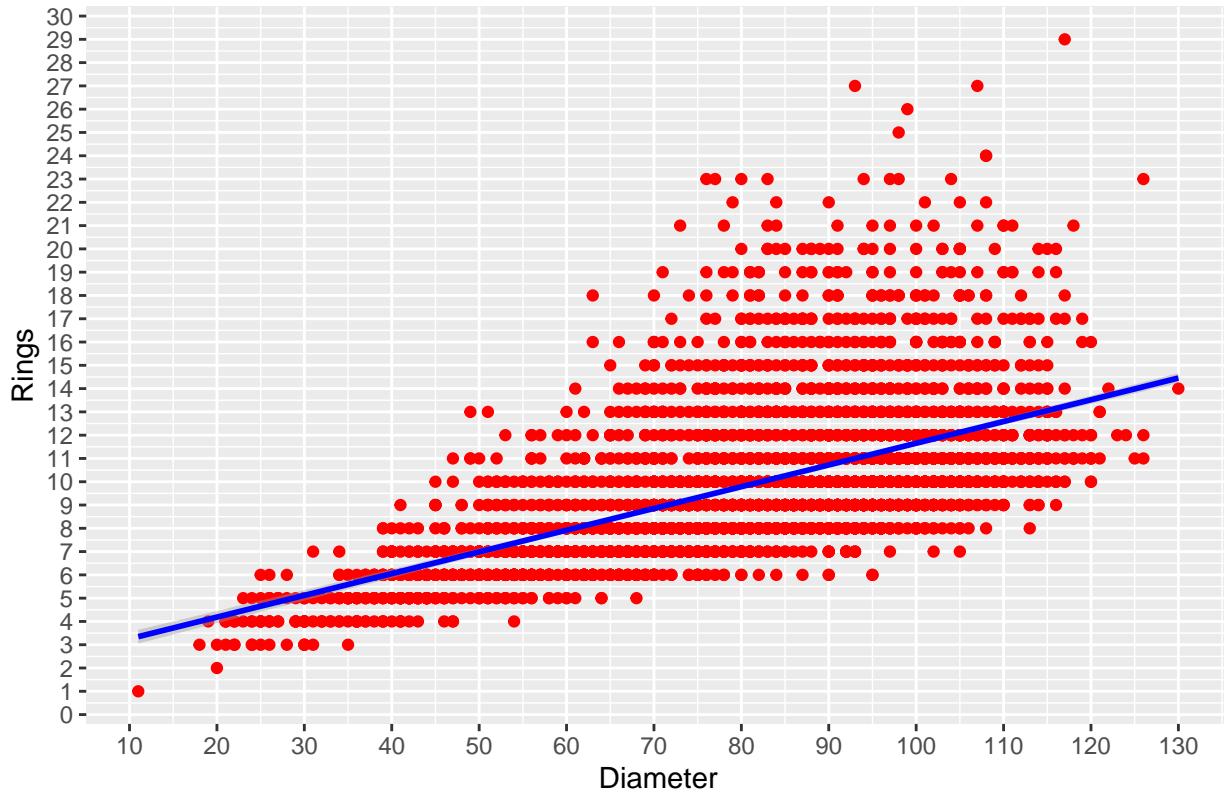
Podemos comprobar que hay bastante heterocedasticidad, en la sección de transformación de datos se explicará que se puede hacer para contrarestar este efecto.

Para el par “Diameter-Rings”:

```
ggplot(abalone,aes(x=Diameter, y=Rings)) + geom_point(color='Red')+ geom_smooth(method=lm, color='Blue')
  labs(title = 'Diagrama de dispersión entre Diameter y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Diameter y Rings



Tenemos un resultado muy similar al anterior, no es de extrañar por la relación estrecha que hay entre estas dos variables.

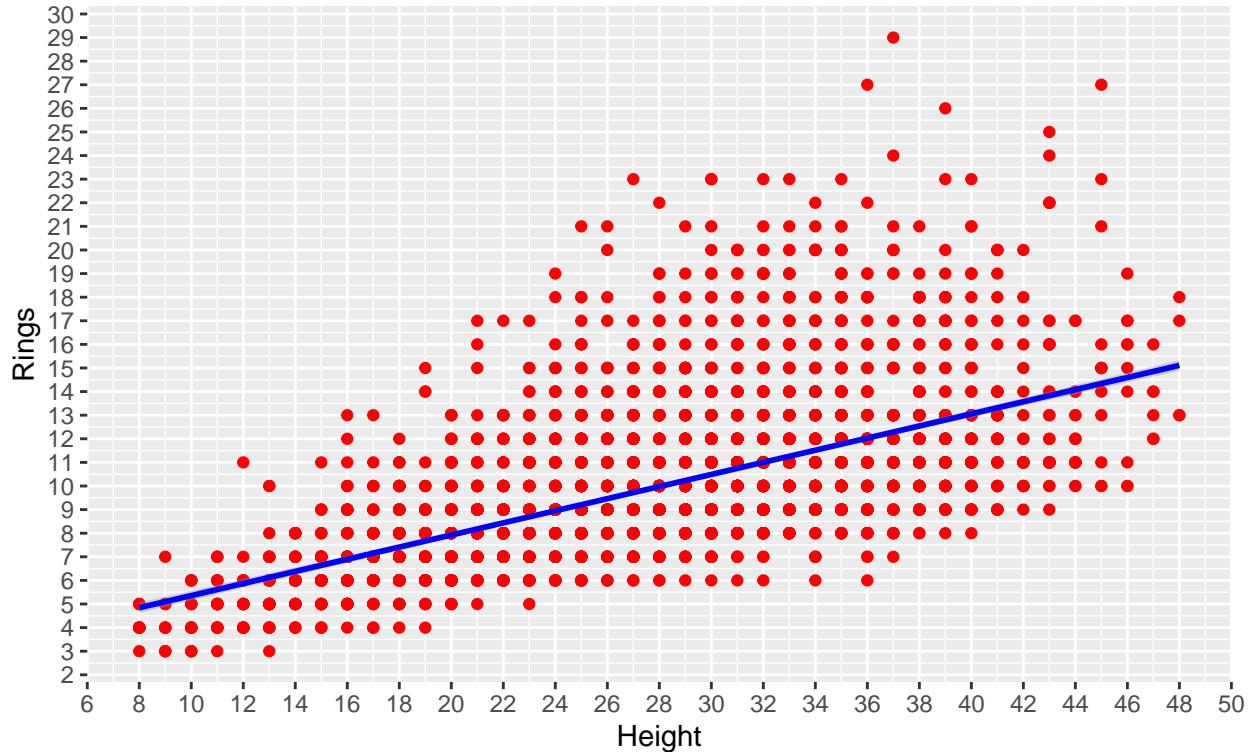
Para el par “Height-Rings” se han eliminado los outliers de “Height” por los valores extremos que dificultarían la visualización.

```
# En este caso si es necesario eliminar los outliers de Height
ggplot(abalone[-c(Height.outliers),],aes(x=Height, y=Rings)) + geom_point(color='Red')+ geom_smooth(method="lm")
  labs(title = 'Diagrama de dispersión entre Height y Rings', subtitle = 'Sin outliers de Height')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=50, by=2))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Height y Rings

Sin outliers de Height



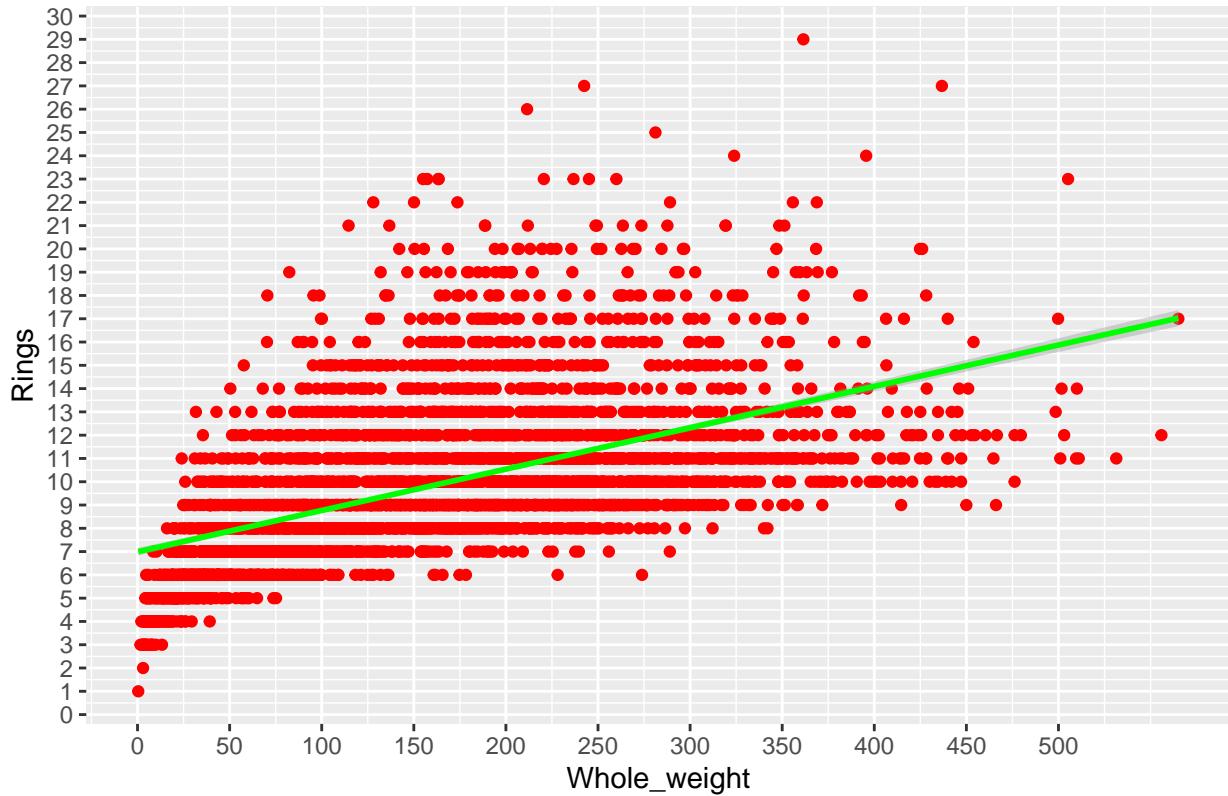
También hay heterocedasticidad, y esto puede dificultar la regresión sobre la variable de salida. Estas diferencias pueden darse por diversos motivos, ya sea por la especie de abulón o el medio en el que se encuentre (temperatura de las aguas, cantidad de alimento disponible, etc.).

Ahora se estudiarán las variables de peso, comenzando por el par “Whole weight-Rings”:

```
# Whole_weight-Rings
ggplot(abalone,aes(x=Whole_weight, y=Rings)) + geom_point(color='Red')+ geom_smooth(method=lm, color='Blue')
  labs(title = 'Diagrama de dispersión entre Whole_weight y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=500, by=50))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Whole_weight y Rings



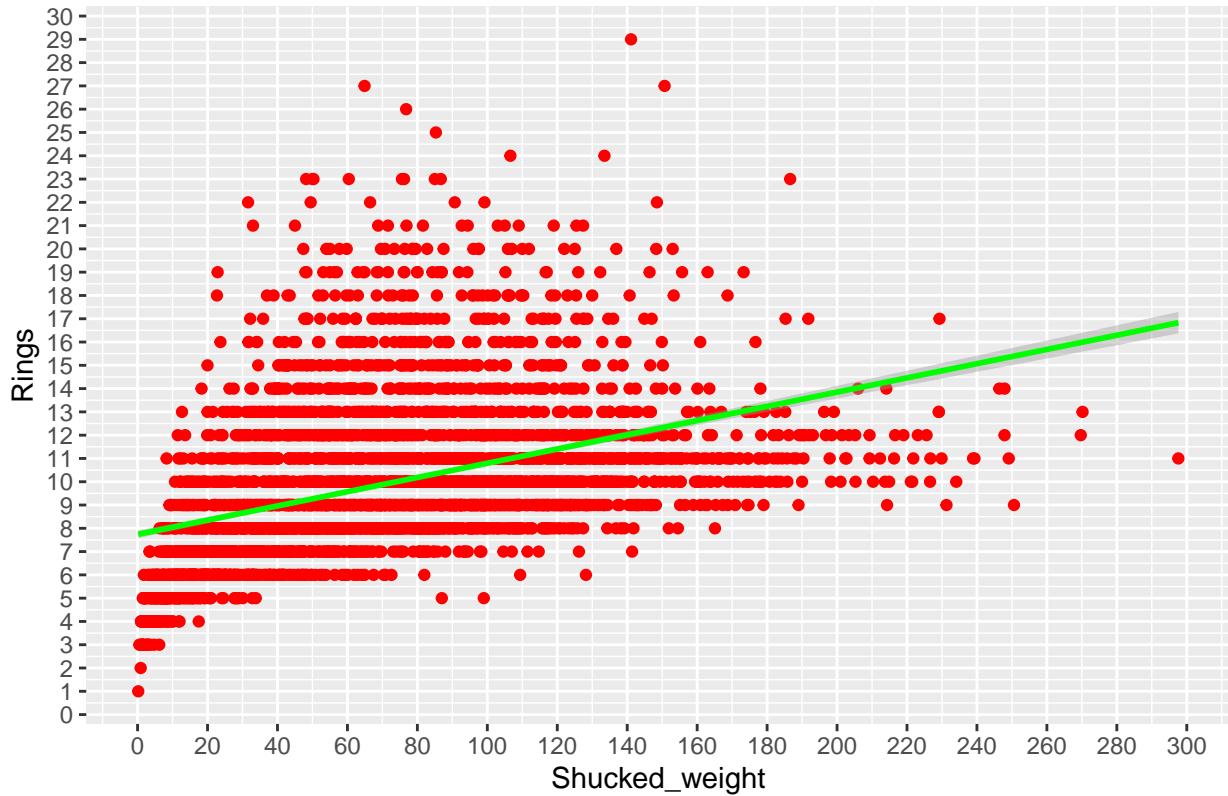
Los datos se distribuyen con una relación que parece logarítmica o cuadrática inversa, en el apartado de transformación de los datos se estudiará esta relación en detalle. Por la relación que se ha explicado anteriormente entre las variables de peso, se espera que las gráficas de las siguientes tres variables no difieran mucho con respecto a la que acabamos de visualizar.

Para el par “Shucked weight-Rings”:

```
# Shucked_weight-Rings
ggplot(abalone,aes(x=Shucked_weight, y=Rings)) + geom_point(color='Red')+  geom_smooth(method=lm, color="green")
  labs(title = 'Diagrama de dispersión entre Shucked_weight y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=300, by=20))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Shucked_weight y Rings



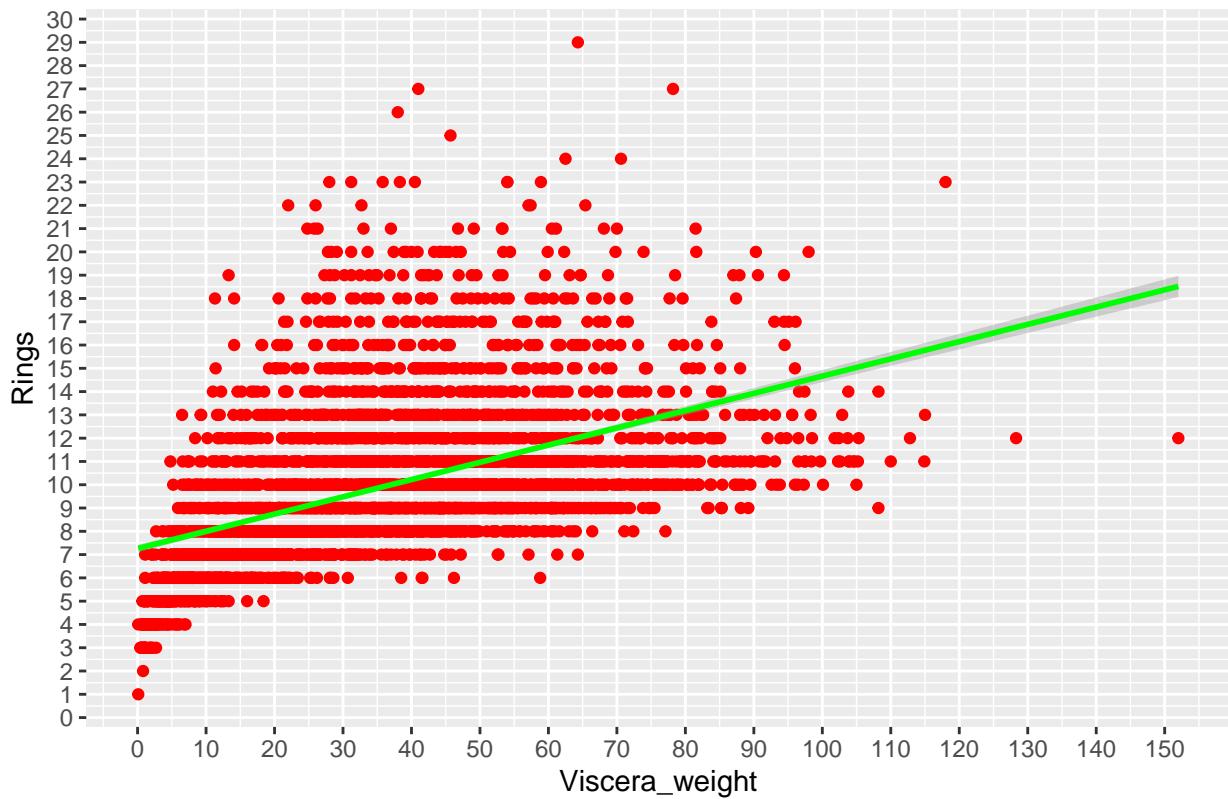
Muy parecida a la anterior, con una dispersión ligeramente diferente.

Para “Viscera weight-Rings”:

```
# Viscera_weight-Rings
ggplot(abalone,aes(x=Viscera_weight, y=Rings)) + geom_point(color='Red')+ geom_smooth(method=lm, color="green")
  labs(title = 'Diagrama de dispersión entre Viscera_weight y Rings')+
  scale_y_continuous(breaks = seq(0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=150, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Viscera_weight y Rings



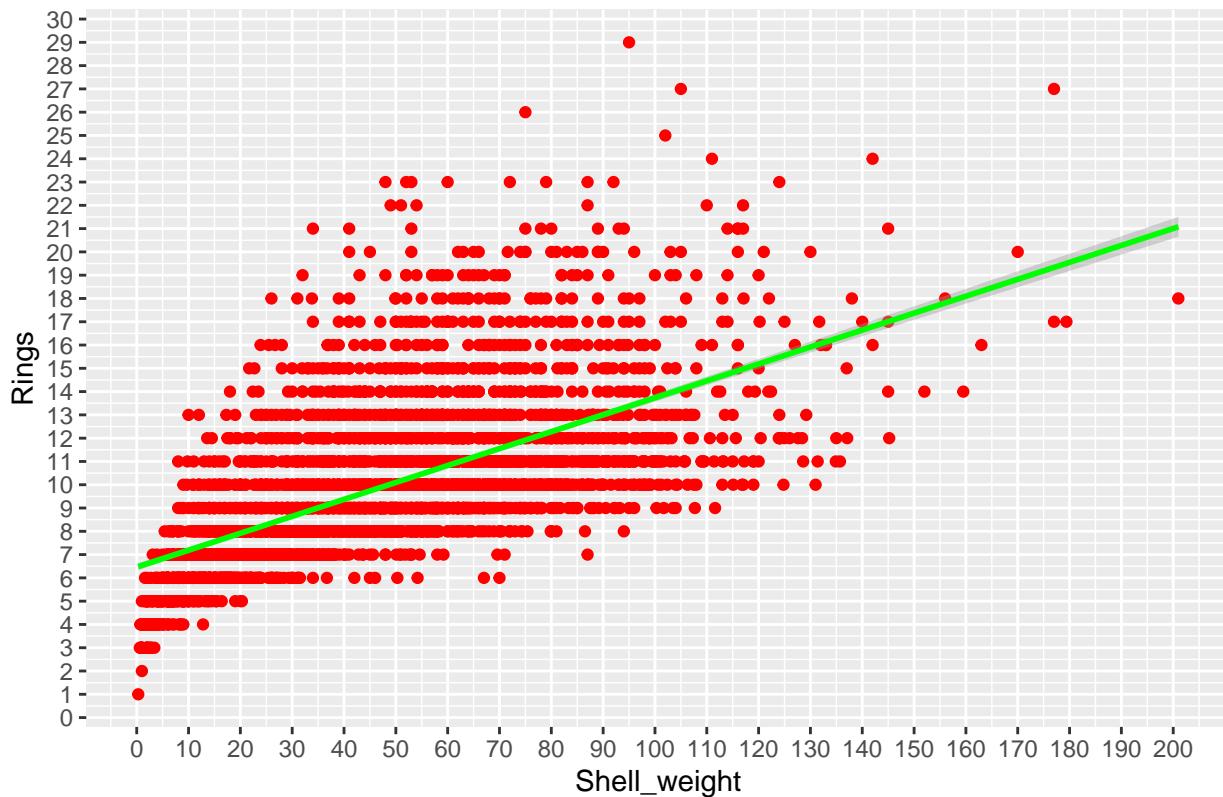
Forma muy parecida a las anteriores como era de esperar.

Y por último, “Shell weight-Rings”:

```
# Shell_weight-Rings
ggplot(abalone,aes(x=Shell_weight, y=Rings)) + geom_point(color='Red')+ geom_smooth(method=lm, color='Green')
  labs(title = 'Diagrama de dispersión entre Shell_weight y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=200, by=10))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Shell_weight y Rings



Forma muy similar a las demás variables de peso, aunque con bastantes puntos dispersos en los valores altos de la variable “Shell weight”.

Descomposición de atributos complicados

A la hora de modelar una regresión es importante considerar y transformar ciertas variables o valores que pueden ser problemáticos. Para este caso particular la variable “Sex” es la única categórica, y en principio al venir descrita mediante números enteros (“1” para “male”, “2” para “female” y “3” para “infant”), no tendríamos ningún conflicto al introducirlo al algoritmo de regresión como entrada. El problema viene que al utilizar enteros estamos introduciendo una noción de “orden” para una variable categórica cuyas categorías no están definidas mediante un orden. Para ello es necesario crear “dummy values”, que no es otra cosa que descomponer esta variable en diferentes columnas según el número de categorías. Ahora nuestras observaciones que pertenecían a cierto grupo, tendrán el valor binario “1” en la columna que corresponde a su categoría y “0” en el resto. Hay que tener cuidado con crear muchas variables, ya que se puede caer en el problema de alta dimensionalidad y algoritmos basados por ejemplo en distancia como knn, se ven afectados negativamente en el sentido que las variables importantes pasan a contribuir menos al modelo cuando tenemos más variables a considerar.

```
# Pasamos la variable categórica a valores binarios
abalone.dummy <- dummy_cols(abalone, remove_selected_columns = TRUE)
head(abalone.dummy)
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight
## 1	80	61	20	68.3	35.2	12.5
## 2	127	100	30	275.2	129.9	72.2
## 3	74	54	18	37.1	14.0	8.5
## 4	136	108	31	306.8	134.2	75.8

```

## 5      75      57      18      50.9      23.8      11.9
## 6    116     95     31    194.8     86.1     46.0
##   Shell_weight Rings Sex_M Sex_F Sex_I
## 1      17.3      7      0      0      1
## 2      62.0     10      0      1      0
## 3      13.0      7      0      0      1
## 4      76.8     10      1      0      0
## 5      13.5      6      0      0      1
## 6      57.0     10      0      1      0

```

Así quedaría el dataset tras la descomposición de la variable categórica en tres diferentes variables.

Búsqueda de datos redundantes

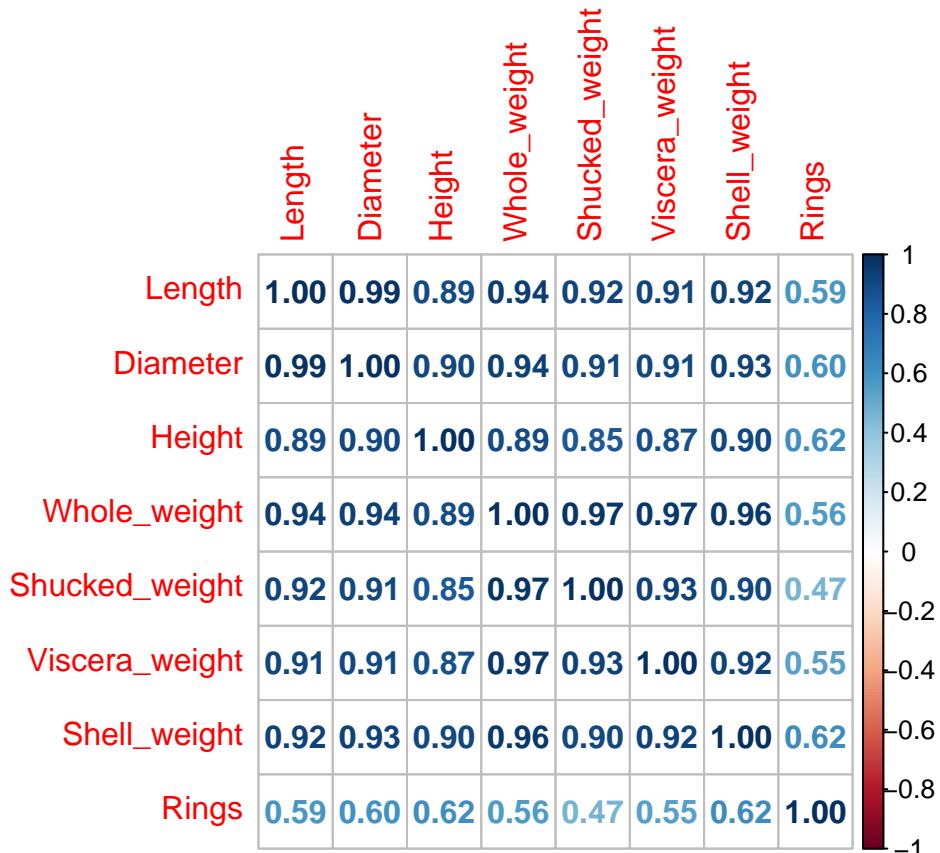
La reducción de dimensionalidad es un aspecto importante a la hora de aplicar un algoritmo de regresión, por un lado nos permite contrarestar el problema de alta dimensionalidad, y por otro lado permite simplificar el modelo seleccionando las variables que mayor influencia tengan en la variable objetivo para realizar inferencias.

Lo primero que podemos hacer es estudiar la correlación que hay entre las variables. Para ello calculamos la matriz de correlación y la representamos. Puesto que de cara al problema se ha decidido eliminar los outliers, estos no se incluirán para calcular la correlación.

```

# Estudiamos la correlación entre las variables mediante su matriz de correlación
abalone.CorrMatrix <- cor(abalone[-c(abalone.outliers), -1])
corrplot(abalone.CorrMatrix, method = "number")

```



Nuestras variables de entrada numéricas están muy correlacionadas. Observando el gráfico se podría determinar las variables a descartar. Por un lado “Length” y “Diameter” tienen una alta correlación, por lo que se puede

prescindir de una de ellas. Por otro lado las variables de peso tienen una correlación muy alta entre ellas, por lo que escogiendo las que menos correlación tienen con la salida, podríamos descartar “Shucked weight” y “Viscera weight”.

No obstante, se podría efectuar un análisis de componentes principales con nuestro conjunto de datos sin outliers para determinar las variables que contribuyen más a los componentes principales.

```
abalone.pca <- PCA(abalone[-abalone.outliers,-c(1,9)], scale.unit = TRUE, ncp = 3, graph = FALSE)
```

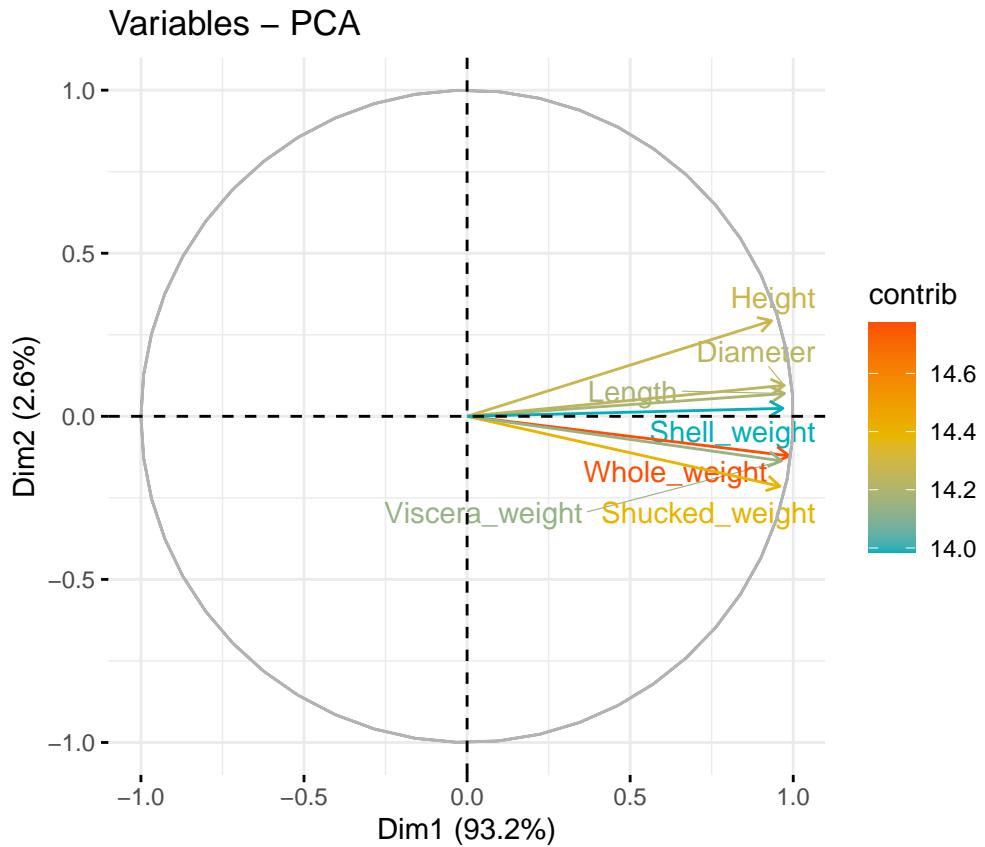
Si extraemos los valores propios:

```
get_eigenvalue(abalone.pca)
```

```
##          eigenvalue variance.percent cumulative.variance.percent
## Dim.1  6.526969043      93.24241490            93.24241
## Dim.2  0.179795014      2.56850021            95.81092
## Dim.3  0.128223417      1.83176309            97.64268
## Dim.4  0.080371289      1.14816128            98.79084
## Dim.5  0.064473677      0.92105253            99.71189
## Dim.6  0.013384871      0.19121244            99.90310
## Dim.7  0.006782689      0.09689555            100.00000
```

Observamos que con una dimensión ya se explica más del 80% de la variabilidad y con dos dimensiones ya se alcanzaría el 95.8%. Si representamos estas dos dimensiones con las variables:

```
fviz_pca_var(abalone.pca,
             axes = c(1, 2),
             col.var = "contrib",
             gradient.cols = c("#00AFBB",
                               "#E7B800", "#FC4E07"),
             repel = TRUE)
```



Si queremos saber la contribución exacta de cada variable en ambas dimensiones:

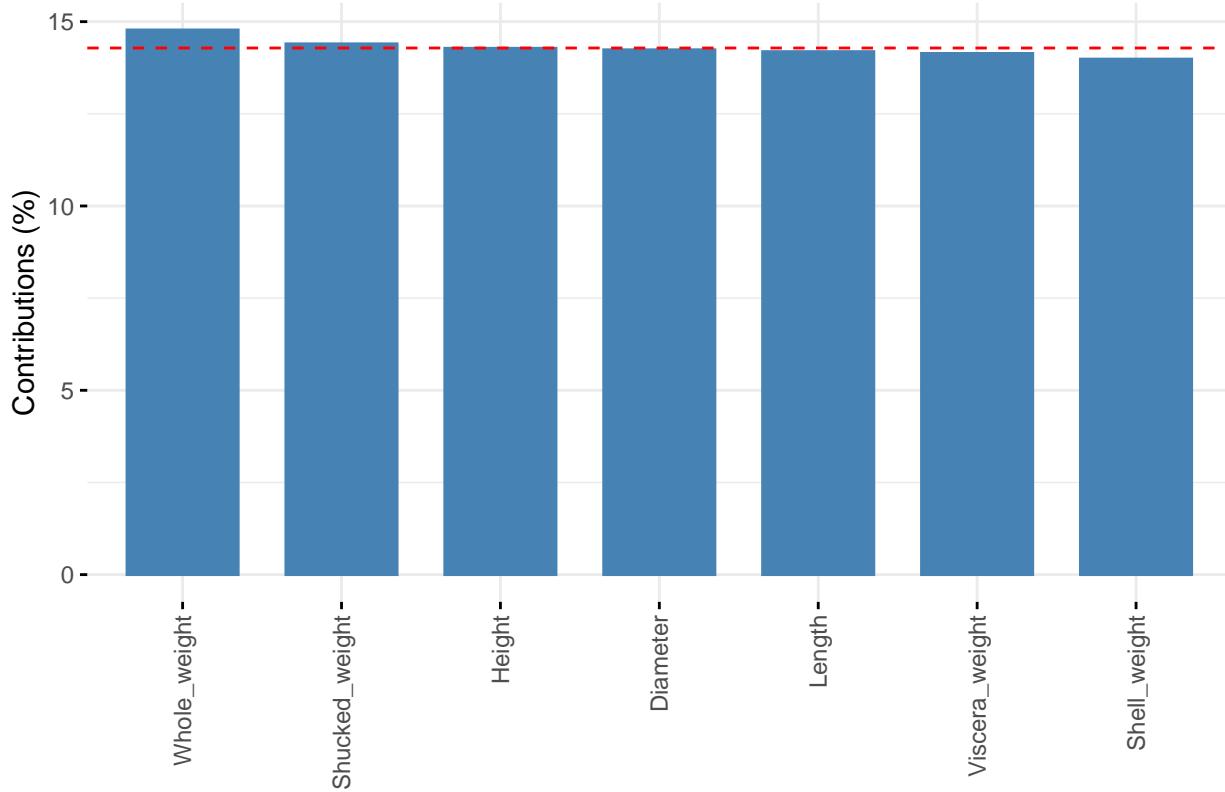
```
get_eigenvalue(abalone.pca)
```

```
##           eigenvalue variance.percent cumulative.variance.percent
## Dim.1  6.526969043      93.24241490            93.24241
## Dim.2  0.179795014      2.56850021            95.81092
## Dim.3  0.128223417      1.83176309            97.64268
## Dim.4  0.080371289      1.14816128            98.79084
## Dim.5  0.064473677      0.92105253            99.71189
## Dim.6  0.013384871      0.19121244            99.90310
## Dim.7  0.006782689      0.09689555          100.00000
```

Representamos la contribución de las variables a ambas dimensiones para visualizar cuáles son las de mayor contribución:

```
fviz_contrib(abalone.pca,
            choice = "var",
            axes = 1:2,
            xtickslab.rt = 90,
            top=8)
```

Contribution of variables to Dim-1–2



En definitiva, con tan sólo dos componentes principales ya se explica un gran porcentaje de la variabilidad de nuestras variables.

Aunque la reducción de dimensionalidad tenga su utilidad, el estudio que hacemos de las variables siguiendo el método de “forward selection” o “backward selection” para la regresión lineal, ya nos permite seleccionar/descartar las variables más importantes para el modelo.

Transformación de datos

La transformación de los datos es un paso importante para asegurarnos que nuestro algoritmo de regresión funcione de forma óptima, es por ello que dependerá del algoritmo que vayamos a utilizar.

Para knn al ser un algoritmo basado en distancias, debemos normalizar los datos.

```
abalone.scaled <- abalone.dummy %>% mutate_if(is.numeric, scale, center = TRUE, scale = TRUE)
head(abalone.scaled)
```

```
##      Length   Diameter    Height Whole_weight Shucked_weight Viscera_weight
## 1 -1.0332757 -1.0374950 -0.9486602 -0.9941985 -0.8266404 -1.0779821
## 2  0.9239277  0.9278372  0.2496504  1.1155220  1.3066815  1.6454230
## 3 -1.2831315 -1.3902470 -1.1883223 -1.3123391 -1.3042162 -1.2604548
## 4  1.2987113  1.3309822  0.3694815  1.4377413  1.4035483  1.8096484
## 5 -1.2414888 -1.2390676 -1.1883223 -1.1716231 -1.0834500 -1.1053530
## 6  0.4658588  0.6758715  0.3694815  0.2956983  0.3199920  0.4502268
##      Shell_weight     Rings     Sex_M     Sex_F     Sex_I
## 1    -1.0942568 -0.91032347 -0.7596835 -0.6749882  1.4543606
## 2     0.5112072  0.02013201 -0.7596835  1.4811525 -0.6874227
## 3    -1.2486974 -0.91032347 -0.7596835 -0.6749882  1.4543606
```

```

## 4    1.0427702  0.02013201  1.3160224 -0.6749882 -0.6874227
## 5   -1.2307391 -1.22047530 -0.7596835 -0.6749882  1.4543606
## 6    0.3316251  0.02013201 -0.7596835  1.4811525 -0.6874227

```

En cuanto a las transformaciones mencionadas en apartados anteriores, para reducir la heterocedasticidad en la variable de salida se podría utilizar la función logarítmica.

Por ejemplo con respecto a la variable “Length” pasariamos de tener:

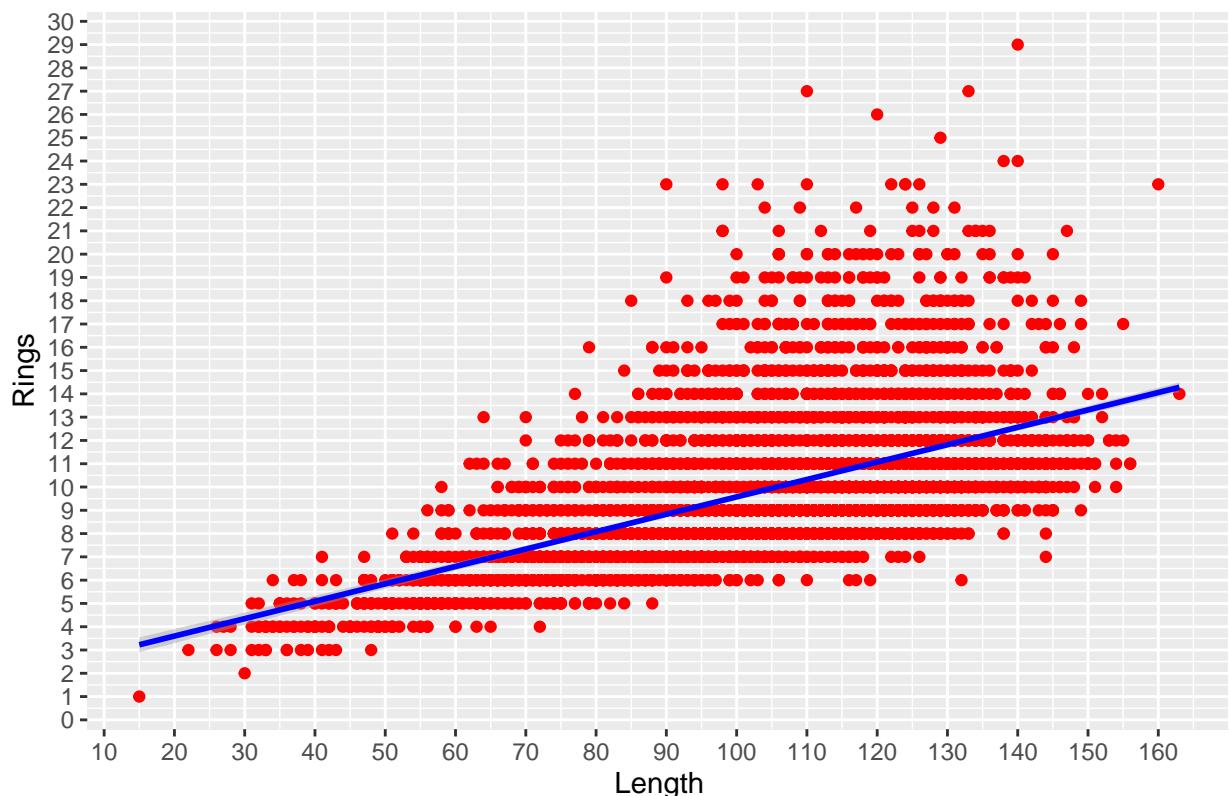
```

# Length-Rings
ggplot(abalone,aes(x=Length, y=Rings)) + geom_point(color='Red')+ 
  geom_smooth(method=lm, color='Blue')+
  labs(title = 'Diagrama de dispersión entre Length y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))

## `geom_smooth()` using formula = 'y ~ x'

```

Diagrama de dispersión entre Length y Rings



A este gráfico de dispersión:

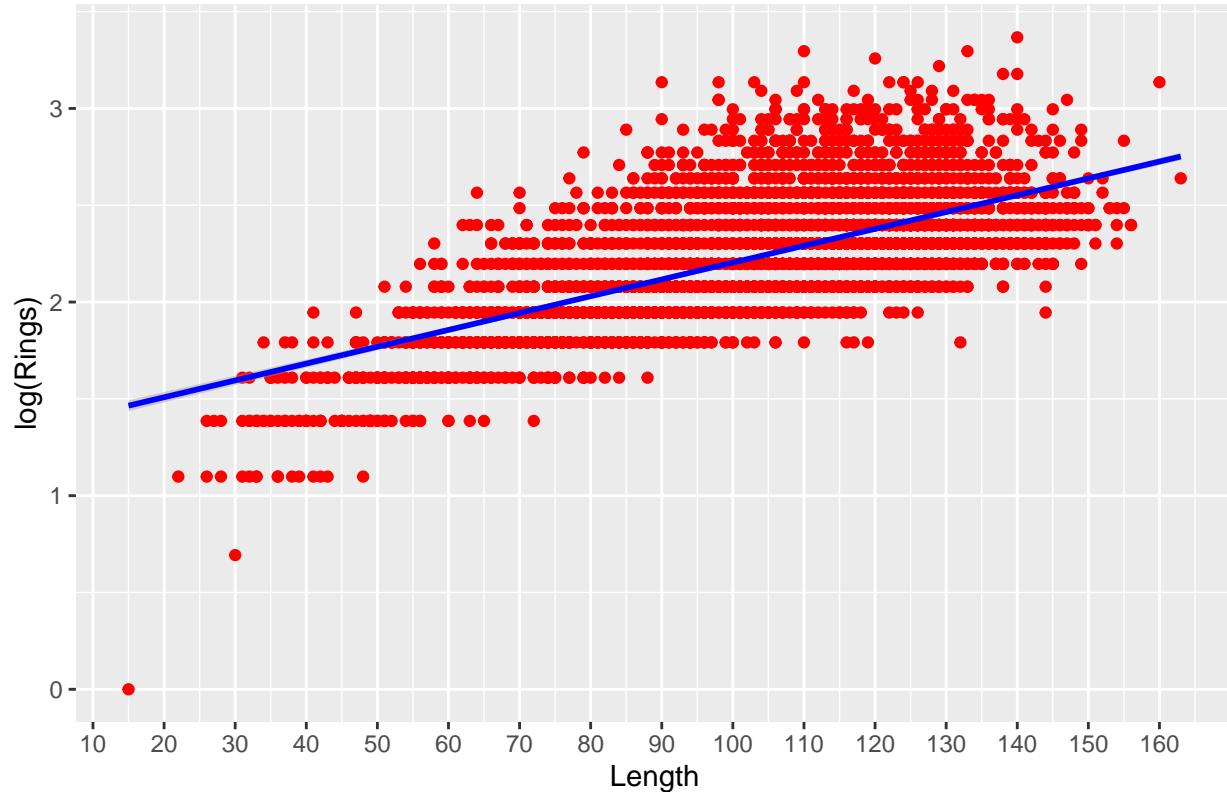
```

# Length-Rings
ggplot(abalone,aes(x=Length, y=log(Rings))) +
  geom_point(color='Red')+ geom_smooth(method=lm, color='Blue')+
  labs(title = 'Diagrama de dispersión entre Length y el logaritmo de Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=160, by=10))

## `geom_smooth()` using formula = 'y ~ x'

```

Diagrama de dispersión entre Length y el logaritmo de Rings



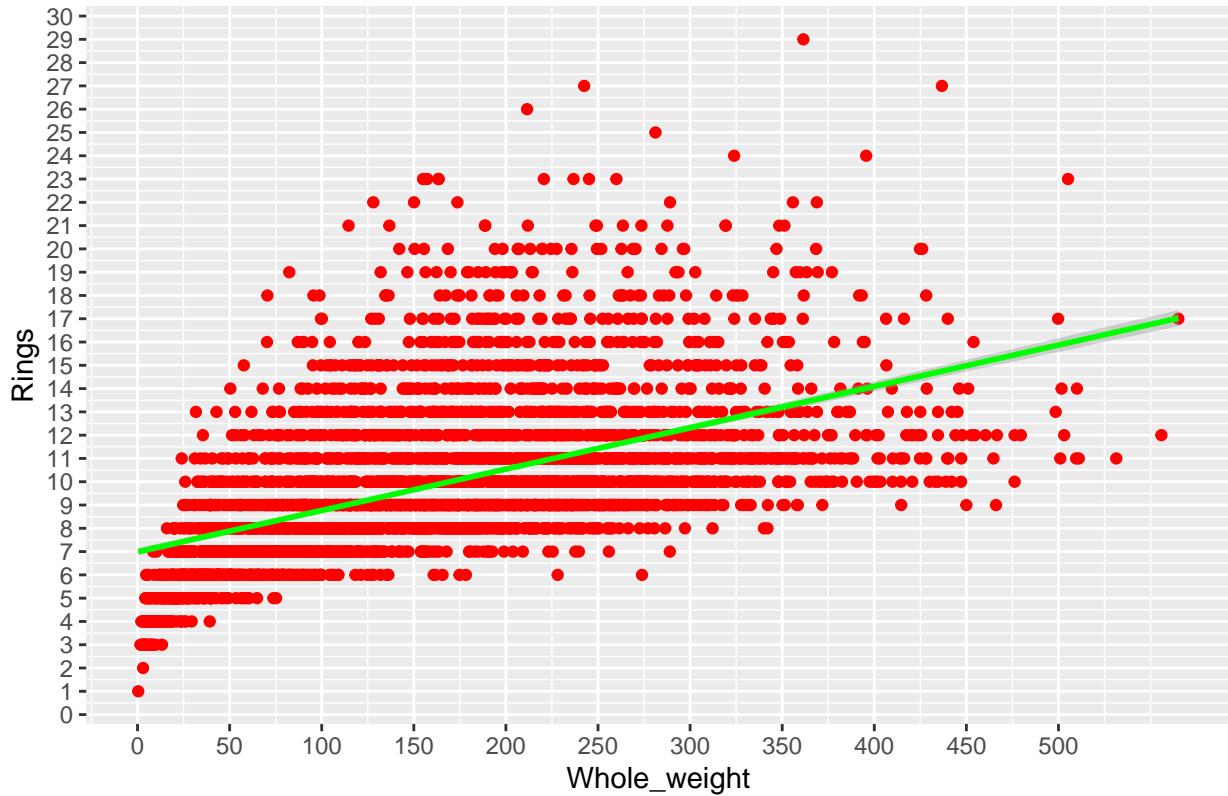
Con esta transformación se consigue una dispersión más homogénea.

En cuanto a la transformación para las variables de peso con respecto a la salida, se ha encontrado que utilizando la raíz cuadrada sobre la variable de entrada funciona bien, pasando de:

```
# Whole_weight-Rings
ggplot(abalone,aes(x=Whole_weight, y=Rings)) + geom_point(color='Red')+
  geom_smooth(method=lm, color='Green')+
  labs(title = 'Diagrama de dispersión entre Whole_weight y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=500, by=50))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre Whole_weight y Rings

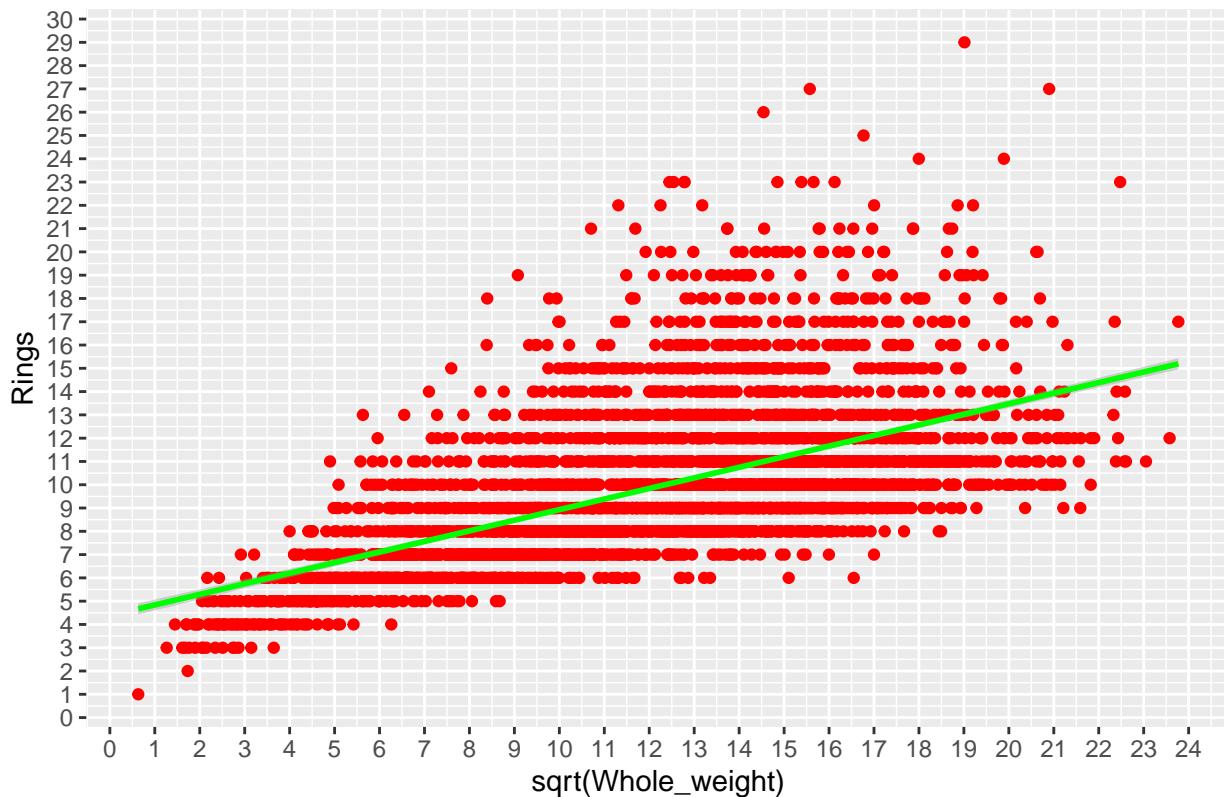


A esto:

```
# Whole_weight-Rings
ggplot(abalone,aes(x=sqrt(Whole_weight), y=Rings)) + geom_point(color='Red')+
  geom_smooth(method=lm, color='Green')+
  labs(title = 'Diagrama de dispersión entre la raíz de Whole_weight y Rings')+
  scale_y_continuous(breaks = seq(from=0, to=30, by=1))+ 
  scale_x_continuous(breaks = seq(from=0, to=25, by=1))

## `geom_smooth()` using formula = 'y ~ x'
```

Diagrama de dispersión entre la raíz de Whole_weight y Rings



Estas dos transformaciones se han seleccionado por mostrar mejores resultados y porque las variables transformadas mostraban un “skewness” positivo.

Conclusiones

Una terminado el análisis exploratorio de los datos, planteamos nuevamente las hipótesis formuladas al principio:

- El tamaño y el peso aumentan con la edad. Si, esto se cumple de forma muy general y es evidente para los individuos de menor tamaño.
- Si es así, ¿Qué relación hay con respecto al número de anillos? Con respecto a la longitud, parece que hay una relación lineal con mucha varianza, en cambio con respecto al peso parece una relación cuadrática inversa.
- ¿Alcanzan estas especies un límite de tamaño y peso? No se ha podido comprobar, ya que parece que hay factores ya nombrados en el análisis que tienen mayor influencia.
- ¿Cómo es la relación entre las variables de dimensión y las de peso? Se vuelve lineal una vez que se tiene en cuenta todas las dimensiones. (longitud x diámetro x altura) ~ peso
- ¿Aumentan todas las variables de tamaño en la misma proporción (están correlacionadas)? Las de longitud y diámetro tienen una enorme correlación, pero con la de altura no hay tanta por tener en cuenta parte de la vianda.
- La variable sexo influye sobre las medidas físicas del individuo. Si tenemos en cuenta la clase infante, a grandes rasgos si.
- ¿Hay dimorfismo sexual? ¿En qué grado? No tenemos evidencia de que la haya, y si la hay es muy pequeña.

La dificultad de este dataset se ha encontrado en el tratamiento de los datos atípicos y las variables que presentaban heterocedasticidad.

Regresión

Una vez realizado el análisis exploratorio de los datos, se procede a realizar el estudio de regresión sobre el dataset.

Lo primero que debemos hacer es cargar los paquetes necesarios y los datos con los que trabajaremos, además de renombrar las variables.

```
# Importamos kknn
require(kknn)

## Loading required package: kknn

## Warning: package 'kknn' was built under R version 4.2.2

# cargamos dataset
abalone.reg <- read.csv("Input/abalone/abalone.dat", comment.char="@", header = FALSE)
# Cambiamos nombres de las variables
n <- length(names(abalone.reg)) - 1
names(abalone.reg)[1:n] <- paste ("X", 1:n, sep="")
names(abalone.reg)[n+1] <- "Y"

str(abalone)

## 'data.frame': 4175 obs. of 9 variables:
## $ Sex : Factor w/ 3 levels "M","F","I": 3 2 3 1 3 2 2 2 1 1 ...
## $ Length : num 80 127 74 136 75 116 105 126 113 121 ...
## $ Diameter : num 61 100 54 108 57 95 76 99 88 93 ...
## $ Height : num 20 30 18 31 18 31 28 38 25 33 ...
## $ Whole_weight : num 68.3 275.2 37.1 306.8 50.9 ...
## $ Shucked_weight: num 35.2 129.9 14 134.2 23.8 ...
## $ Viscera_weight: num 12.5 72.2 8.5 75.8 11.9 46 29.5 42.3 36.5 49.5 ...
## $ Shell_weight : num 17.3 62 13 76.8 13.5 57 42 32.5 43 68 ...
## $ Rings : int 7 10 7 10 6 10 14 10 9 13 ...
```

Obtención de modelos de regresión lineal simple

Para esta primera sección es necesario escoger cinco de los regresores disponibles que consideremos más relevantes. Fijándonos en el estudio con respecto a la variable de salida y en la matriz de correlación calculada de los atributos numéricos en la parte de análisis exploratorio, se podría considerar que los atributos menos relevantes son: por un lado la variable categórica “Sex”, que debido a su naturaleza no es muy útil para calcular una regresión con la variable de salida; y por otro lado, las variables “Shucked weight” y “Viscera weight” que tienen la menos correlación con la salida.

Con esto en mente, se construyen los cinco regresores simples:

```
# 1. Obtención de modelos lineales simples

# X2 (Length):
fitX2=lm(Y~X2,data=abalone.reg)
summary(fitX2)

## 
## Call:
## lm(formula = Y ~ X2, data = abalone.reg)
```

```

## 
## Residuals:
##   Min     1Q Median     3Q    Max
## -5.9665 -1.6961 -0.7423  0.8733 16.6776
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.1019    0.1855   11.33 <2e-16 ***
## X2          14.9464    0.3452   43.30 <2e-16 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.679 on 4175 degrees of freedom
## Multiple R-squared:  0.3099, Adjusted R-squared:  0.3098 
## F-statistic: 1875 on 1 and 4175 DF,  p-value: < 2.2e-16

# X3 (Diameter):
fitX3=lm(Y~X3,data=abalone.reg)
summary(fitX3)

## 
## Call:
## lm(formula = Y ~ X3, data = abalone.reg)
## 
## Residuals:
##   Min     1Q Median     3Q    Max
## -5.1868 -1.6932 -0.7200  0.9066 15.9999
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  2.3186    0.1727   13.42 <2e-16 ***
## X3          18.6699    0.4115   45.37 <2e-16 ***
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.639 on 4175 degrees of freedom
## Multiple R-squared:  0.3302, Adjusted R-squared:  0.3301 
## F-statistic: 2059 on 1 and 4175 DF,  p-value: < 2.2e-16

# X4 (Height):
fitX4=lm(Y~X4,data=abalone.reg)
summary(fitX4)

## 
## Call:
## lm(formula = Y ~ X4, data = abalone.reg)
## 
## Residuals:
##   Min     1Q Median     3Q    Max
## -44.496 -1.657 -0.607  0.839 17.112
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  3.9385    0.1443   27.30 <2e-16 ***
## X4          42.9714    0.9904   43.39 <2e-16 ***

```

```

## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.677 on 4175 degrees of freedom
## Multiple R-squared: 0.3108, Adjusted R-squared: 0.3106
## F-statistic: 1882 on 1 and 4175 DF, p-value: < 2.2e-16

# X5 (Whole_weight):
fitX5=lm(Y~X5,data=abalone.reg)
summary(fitX5)

##
## Call:
## lm(formula = Y ~ X5, data = abalone.reg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6.2693 -1.7518 -0.6874  1.0177 15.7029
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.98924   0.08244  84.78  <2e-16 ***
## X5          3.55291   0.08562  41.50  <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.713 on 4175 degrees of freedom
## Multiple R-squared: 0.292, Adjusted R-squared: 0.2919
## F-statistic: 1722 on 1 and 4175 DF, p-value: < 2.2e-16

# X8 (Shell_weight):
fitX8=lm(Y~X8,data=abalone.reg)
summary(fitX8)

##
## Call:
## lm(formula = Y ~ X8, data = abalone.reg)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -5.9830 -1.6005 -0.5843  0.9390 15.6334
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.46212   0.07715  83.76  <2e-16 ***
## X8          14.53568   0.27908  52.08  <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.51 on 4175 degrees of freedom
## Multiple R-squared: 0.3938, Adjusted R-squared: 0.3937
## F-statistic: 2713 on 1 and 4175 DF, p-value: < 2.2e-16

```

Se obtienen los resultados mostrados en la siguiente tabla:

Regresor	R cuadrado
X2 (Length)	0.3099
X3 (Diameter)	0.3302
X4 (Height)	0.3108
X5 (Whole weight)	0.292
X8 (Shell weight)	0.3938

Según los resultados obtenidos, la variable “Shell weight” es la que mejores métricas ha obtenido no es de extrañar puesto que era de las que más correlación tenía con la salida. Por curiosidad, si aplicamos la transformación propuesta (raíz cuadrada) a la variable de entrada, obtendríamos el siguiente resultado:

```
fitX8.sqrt=lm(Y-sqrt(X8), data=abalone.reg)
summary(fitX8.sqrt)
```

```
##
## Call:
## lm(formula = Y ~ sqrt(X8), data = abalone.reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -5.6882 -1.5544 -0.5982  0.8651 15.9750 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  3.5435    0.1222   28.99 <2e-16 ***
## sqrt(X8)    13.7573    0.2501   55.01 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.455 on 4175 degrees of freedom
## Multiple R-squared:  0.4202, Adjusted R-squared:  0.4201 
## F-statistic: 3026 on 1 and 4175 DF, p-value: < 2.2e-16
```

Se mejora el modelo obteniéndose un R cuadrado de 0.4202.

Obtención de modelos de regresión lineal múltiple

En esta sección el objetivo es encontrar un modelo de regresión lineal múltiple que describa la variable de salida de forma más precisa que el mejor modelo de regresión lineal simple obtenido. Para ello se deben seguir varios pasos, comenzando por encontrar el subconjunto de variables que sean más útiles, después buscar términos de interacción y por último introducir componentes no lineales.

Para la selección de variables se ha seguido el método de “backward selection”, que consiste en empezar con todas las variables en el modelo e ir descartando variables una a una según el p-valor obtenido.

Con todas las variables tenemos el siguiente modelo:

```
# Con todas las variables:
fit1=lm(Y~, abalone.reg)
summary(fit1)
```

```
##
## Call:
## lm(formula = Y ~ ., data = abalone.reg)
##
## Residuals:
```

```

##      Min     1Q   Median     3Q     Max
## -10.5991 -1.3120 -0.3549  0.8968 14.0582
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.1478    0.3013 13.765 < 2e-16 ***
## X1          -0.3885   0.0467 -8.319 < 2e-16 ***
## X2          -0.8264   1.8122 -0.456   0.648
## X3          11.9640   2.2254  5.376 8.02e-08 ***
## X4          11.2045   1.5374  7.288 3.75e-13 ***
## X5          9.0702    0.7270 12.476 < 2e-16 ***
## X6         -20.1061   0.8168 -24.617 < 2e-16 ***
## X7          -10.1551   1.2941 -7.847 5.36e-15 ***
## X8          8.7011    1.1277  7.716 1.49e-14 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.2 on 4168 degrees of freedom
## Multiple R-squared: 0.5353, Adjusted R-squared: 0.5345
## F-statistic: 600.3 on 8 and 4168 DF, p-value: < 2.2e-16

```

En este primer paso observamos que la variable con mayor p-valor es la variable X2 (Length), por lo que decidí retirarla y volver a calcular.

```

fit2=lm(Y~.-X2, abalone.reg)
summary(fit2)

```

```

##
## Call:
## lm(formula = Y ~ . - X2, data = abalone.reg)
##
## Residuals:
##      Min     1Q   Median     3Q     Max
## -10.5714 -1.3106 -0.3511  0.8918 14.0850
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.10391   0.28555 14.372 < 2e-16 ***
## X1          -0.38954   0.04664 -8.352 < 2e-16 ***
## X3          11.05438   0.98630 11.208 < 2e-16 ***
## X4          11.18384   1.53662  7.278 4.02e-13 ***
## X5          9.07432    0.72691 12.483 < 2e-16 ***
## X6         -20.13583   0.81408 -24.734 < 2e-16 ***
## X7         -10.20929   1.28847 -7.924 2.94e-15 ***
## X8          8.71714    1.12699  7.735 1.29e-14 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.2 on 4169 degrees of freedom
## Multiple R-squared: 0.5353, Adjusted R-squared: 0.5345
## F-statistic: 686.1 on 7 and 4169 DF, p-value: < 2.2e-16

```

Con el nuevo modelo los valores de R cuadrado y R cuadrado ajustado no han bajado, ya que la variable descartada tenía un p-valor mayor a 0.05.

Aquí ya tendríamos las variables más útiles, pero podemos probar quitar la de mayor p-valor y ver qué sucede.

```

fit3=lm(Y~.-X2-X4, abalone.reg)
summary(fit3)

##
## Call:
## lm(formula = Y ~ . - X2 - X4, data = abalone.reg)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.2669 -1.3216 -0.3672  0.8892 13.8981
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.47808   0.28262 15.845 < 2e-16 ***
## X1          -0.40557   0.04688 -8.652 < 2e-16 ***
## X3          13.29679   0.94277 14.104 < 2e-16 ***
## X5          9.17316   0.73130 12.544 < 2e-16 ***
## X6         -20.32816   0.81871 -24.830 < 2e-16 ***
## X7          -9.73483   1.29481 -7.518 6.75e-14 ***
## X8          9.57291   1.12781  8.488 < 2e-16 ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.213 on 4170 degrees of freedom
## Multiple R-squared:  0.5294, Adjusted R-squared:  0.5287
## F-statistic: 781.9 on 6 and 4170 DF, p-value: < 2.2e-16

```

Los valores de R cuadrado y R cuadrado ajustado han bajado muy poco y se ha podido descartar la variable X4 (Height), podemos aceptar esta pequeña pérdida a cambio de ganar simplicidad del modelo. Decido quedarme con este modelo y paso a estudiar términos de interacción. Cabe destacar que por el principio de jerarquía, las variables escogidas no se van a modificar aunque se introduzcan términos de interacción que hagan que el p-valor de la variable baje.

Podemos probar relacionar la variable “Diameter” con la de sexo, aunque según lo que se ha estudiado no hay una relación grande.

```

fit4=lm(Y~.-X2-X4+X1*X3, abalone.reg)
summary(fit4)

##
## Call:
## lm(formula = Y ~ . - X2 - X4 + X1 * X3, data = abalone.reg)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.3306 -1.2990 -0.3387  0.8629 13.7831
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.9127    0.5615 14.092 < 2e-16 ***
## X1          -1.7294   0.1931 -8.956 < 2e-16 ***
## X3          3.7618    1.6432  2.289  0.0221 *
## X5          9.3900    0.7277 12.904 < 2e-16 ***
## X6         -19.8561   0.8167 -24.313 < 2e-16 ***
## X7          -9.6301   1.2874 -7.480 8.98e-14 ***

```

```

## X8          9.8441    1.1219   8.774 < 2e-16 ***
## X1:X3      3.4347    0.4862   7.065 1.88e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.2 on 4169 degrees of freedom
## Multiple R-squared:  0.535, Adjusted R-squared:  0.5342
## F-statistic: 685.2 on 7 and 4169 DF, p-value: < 2.2e-16

```

Las métricas suben un poco, aunque puede ser interesante incluir información que relacione el tamaño con el peso. Pruebo introducir un término de interacción entre las variables “Diameter” y “Shell weight”.

```

fit5=lm(Y~.-X2-X4+X3*X8, abalone.reg)
summary(fit5)

```

```

##
## Call:
## lm(formula = Y ~ . - X2 - X4 + X3 * X8, data = abalone.reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -9.2522 -1.3045 -0.3505  0.8654 15.1147 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  4.09227   0.27952 14.640 < 2e-16 ***
## X1          -0.30959   0.04673 -6.625 3.91e-11 ***
## X3          9.08942   0.98839  9.196 < 2e-16 ***
## X5         10.40017   0.72562 14.333 < 2e-16 ***
## X6        -19.63672   0.80652 -24.347 < 2e-16 ***
## X7         -9.38533   1.27272 -7.374 1.98e-13 ***
## X8         31.78516   2.12940 14.927 < 2e-16 ***
## X3:X8     -42.93388   3.51449 -12.216 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.175 on 4169 degrees of freedom
## Multiple R-squared:  0.5457, Adjusted R-squared:  0.5449
## F-statistic: 715.3 on 7 and 4169 DF, p-value: < 2.2e-16

```

Tenemos una mejora notable con esta interacción. Probamos ahora incluir una interacción entre las dos variables de peso.

```

# Comprobamos términos de interacción entre las variables de peso
fit6=lm(Y~.-X2-X4+X3*X8+X5*X6, abalone.reg)
summary(fit6)

```

```

##
## Call:
## lm(formula = Y ~ . - X2 - X4 + X3 * X8 + X5 * X6, data = abalone.reg)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -9.6332 -1.2992 -0.3301  0.8624 15.2801 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  4.09227   0.27952 14.640 < 2e-16 ***
## X1          -0.30959   0.04673 -6.625 3.91e-11 ***
## X3          9.08942   0.98839  9.196 < 2e-16 ***
## X5         10.40017   0.72562 14.333 < 2e-16 ***
## X6        -19.63672   0.80652 -24.347 < 2e-16 ***
## X7         -9.38533   1.27272 -7.374 1.98e-13 ***
## X8         31.78516   2.12940 14.927 < 2e-16 ***
## X3:X8     -42.93388   3.51449 -12.216 < 2e-16 ***
## X5:X6     -1.29920   0.86240 -1.510 0.13000    
## X3:X5     -0.33010   0.86240 -0.382 0.70400    
## X3:X6     -19.63672   0.86240 -22.620 < 2e-16 ***
## X5:X8     -42.93388   0.86240 -49.500 < 2e-16 ***
## X6:X8      31.78516   0.86240  37.180 < 2e-16 ***
## X3:X6:X8 -42.93388   0.86240 -49.500 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.175 on 4169 degrees of freedom
## Multiple R-squared:  0.5457, Adjusted R-squared:  0.5449
## F-statistic: 715.3 on 7 and 4169 DF, p-value: < 2.2e-16

```

```

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.73598   0.32855  8.327 < 2e-16 ***
## X1          -0.33036   0.04648 -7.107 1.39e-12 ***
## X3          16.17854   1.34442 12.034 < 2e-16 ***
## X5           8.53165   0.76018 11.223 < 2e-16 ***
## X6         -24.17103   0.99337 -24.332 < 2e-16 ***
## X7          -9.01607   1.26478 -7.129 1.19e-12 ***
## X8          42.81365   2.55233 16.774 < 2e-16 ***
## X3:X8      -64.76167   4.49254 -14.415 < 2e-16 ***
## X5:X6       3.42672   0.44410  7.716 1.49e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.16 on 4168 degrees of freedom
## Multiple R-squared:  0.5521, Adjusted R-squared:  0.5512
## F-statistic: 642.2 on 8 and 4168 DF,  p-value: < 2.2e-16

```

Obtenemos algo de mejora, decidí no incluir este término de interacción ya que complica el modelo y no se gana tanto en términos de mejora de R cuadrado y R cuadrado ajustado.

Podríamos probar incluir la interacción como una división entre las variables de peso que hemos probado. Como hemos visto anteriormente, X5 (Whole weight) estaba muy relacionada con la suma de las otras tres variables de peso, por lo que con este término nos puede dar información sobre la proporción entre el peso total y el peso de la vianda.

```

fit7=lm(Y~.-X2-X4+X3*X8+I(X5/X6), abalone.reg)
summary(fit7)

```

```

##
## Call:
## lm(formula = Y ~ . - X2 - X4 + X3 * X8 + I(X5/X6), data = abalone.reg)
##
## Residuals:
##      Min      1Q      Median      3Q      Max
## -10.2823  -1.2835  -0.3139   0.8731  15.3483
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.31225   0.59247 -3.903 9.66e-05 ***
## X1          -0.22326   0.04646 -4.805 1.60e-06 ***
## X3          11.51216   0.99141 11.612 < 2e-16 ***
## X5           5.17211   0.83194  6.217 5.57e-10 ***
## X6          -8.02729   1.23841 -6.482 1.01e-10 ***
## X7          -9.63167   1.25089 -7.700 1.69e-14 ***
## X8          30.39469   2.09572 14.503 < 2e-16 ***
## I(X5/X6)     2.35684   0.19318 12.201 < 2e-16 ***
## X3:X8      -40.98456   3.45748 -11.854 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.137 on 4168 degrees of freedom
## Multiple R-squared:  0.5613, Adjusted R-squared:  0.5605
## F-statistic: 666.7 on 8 and 4168 DF,  p-value: < 2.2e-16

```

Mejora más que el anterior

Este modelo mejora considerablemente con respecto a los dos anteriores con un valor de R cuadrado de 0.5613, por lo que decido quedarme con este.

Ahora es el momento de incluir algunas no linealidades, podemos probar utilizar la raíz cuadrada o el logaritmo para corregir el “skewness” en las variables de peso.

```
# Añadimos términos no lineales
fit8=lm(Y~.-X2-X4+X3*X8+I(X5/X6)+I(sqrt(X8)), abalone.reg)
summary(fit8)

##
## Call:
## lm(formula = Y ~ . - X2 - X4 + X3 * X8 + I(X5/X6) + I(sqrt(X8)),
##      data = abalone.reg)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -10.1832 -1.2977 -0.3064  0.8627 14.9595 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.53386   0.59239 -4.277 1.93e-05 ***
## X1          -0.22173   0.04633 -4.786 1.76e-06 ***
## X3          -2.69386   2.99196 -0.900   0.368    
## X5          5.57537   0.83338  6.690 2.53e-11 ***
## X6         -8.86250   1.24592 -7.113 1.33e-12 ***
## X7        -10.00433   1.24946 -8.007 1.51e-15 ***
## X8        -11.48978   8.58420 -1.338   0.181    
## I(X5/X6)    2.20625   0.19493 11.318 < 2e-16 ***
## I(sqrt(X8)) 25.34644   5.03848  5.031 5.10e-07 ***
## X3:X8      1.15708   9.05873  0.128   0.898    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.131 on 4167 degrees of freedom
## Multiple R-squared:  0.564, Adjusted R-squared:  0.5631 
## F-statistic: 598.9 on 9 and 4167 DF, p-value: < 2.2e-16
# No mejora mucho
```

Hay una leve mejoría en los valores de R cuadrado y el ajustado. Pruebo ahora con el logaritmo de la variable X6 (Shucked weight).

```
fit9=lm(Y~.-X2-X4+X3*X8+I(X5/X6)+I(log(X6)), abalone.reg)
summary(fit9)

##
## Call:
## lm(formula = Y ~ . - X2 - X4 + X3 * X8 + I(X5/X6) + I(log(X6)),
##      data = abalone.reg)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -10.3247 -1.2779 -0.3114  0.8406 15.3112 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -2.53386   0.59239 -4.277 1.93e-05 ***
```

```

## (Intercept) 3.0172    0.9516   3.171 0.001531 ***
## X1          -0.2306    0.0462  -4.992 6.22e-07 ***
## X3          0.8864    1.7866   0.496 0.619821
## X5          3.2668    0.8691   3.759 0.000173 ***
## X6          -6.7185    1.2447  -5.398 7.12e-08 ***
## X7          -9.0880    1.2458  -7.295 3.56e-13 ***
## X8          16.9593    2.8089   6.038 1.70e-09 ***
## I(X5:X6)    3.2911    0.2325  14.157 < 2e-16 ***
## I(log(X6))  1.6017    0.2246   7.131 1.17e-12 ***
## X3:X8     -12.8023    5.2377  -2.444 0.014556 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.125 on 4167 degrees of freedom
## Multiple R-squared:  0.5666, Adjusted R-squared:  0.5657
## F-statistic: 605.4 on 9 and 4167 DF,  p-value: < 2.2e-16
# mejora poco

```

Hay algo más de mejora que utilizando la raíz cuadrada, pero considero que el valor es muy pequeño como para incluir este término, ya que complica el modelo.

Decido quedarme con el modelo “fit7” entonces.

Aplicar k-nn y el mejor modelo de regresión múltiple obtenido con validación cruzada

Una vez encontrado un modelo que tenga buenas métricas pasamos a compararlo con el algoritmo de “k-nearest neighbor”. Para ellos los dos modelos se evaluarán utilizando las particiones de los datos de entrenamiento y de evaluación proporcionadas.

Para compararlos se utilizará la media del error cuadrático medio obtenido en las cinco particiones. Empezando por knn tenemos:

```

nombre <- "Input/abalone/abalone"

#----- 5-fold cross-validation KNN todas las variables

run_knn_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@", header=FALSE)
  file <- paste(x, "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@", header=FALSE)
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
  }
  fitMulti=kknn(Y~.,x_tra,test)
  yprime=fitMulti$fitted.values
}

```

```

    sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
knnMSEtrain<-mean(sapply(1:5,run_knn_fold,nombre,"train"))
knnMSEtest<-mean(sapply(1:5,run_knn_fold,nombre,"test"))

```

Y se obtienen los siguientes resultados:

```

# Mostramos resultados
print('Resultados modelo knn con todas las variables')

## [1] "Resultados modelo knn con todas las variables"
print(knnMSEtrain)

## [1] 2.217759
print(knnMSEtest)

## [1] 5.398169

```

En cuanto al mejor modelo obtenido, ejecutamos con una función similar:

```

----- 5-fold cross-validation LM de las variables seleccionadas

run_lm_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@", header=FALSE)
  file <- paste(x, "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@", header=FALSE)
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
  }
  fitMulti=lm(Y~.-X2-X4+X3*X8+I(X5/X6),x_tra)
  yprime=predict(fitMulti,test)
  sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))

```

Obteniéndose los siguientes resultados:

```

# Mostramos resultados
print('Resultados modelo lineal con las variables seleccionadas')

## [1] "Resultados modelo lineal con las variables seleccionadas"
print(lmMSEtrain)

## [1] 4.55601

```

```
print(lmMSEtest)
```

```
## [1] 4.584297
```

Curiosamente para los subconjuntos de entrenamiento, el algoritmo que ha obtenido un menor error ha sido knn con un valor de 2.217759 contra el valor de 4.55601 del modelo de regresión lineal múltiple. Pero si comparamos los resultados obtenidos en el subconjunto de “test”, nuestro modelo obtiene un valor de error menor que knn, con un valor de 4.584297 para regresión lineal múltiple y 5.398169 para knn. Por lo que podemos concluir que el modelo obtenido tiene mayor capacidad de generalización al haber obtenido mejores métricas con los datos nuevos del subconjunto de “test”.

Comparativa de algoritmos de regresión múltiple

Para esta última parte, el objetivo es realizar una regresión lineal múltiple con todos los regresores y aplicando validación cruzada, e incluir los resultados tanto de este experimento como el de knn, en la tabla proporcionada. Una vez incluidos, se hace un estudio comparativo de los distintos algoritmos de regresión.

Como se ha hecho anteriormente, aplicamos regresión lineal múltiple, pero esta vez sin quitar variables ni incluir términos nuevos.

```
run_lm_fold <- function(i, x, tt = "test") {  
  file <- paste(x, "-5-", i, "tra.dat", sep="")  
  x_tra <- read.csv(file, comment.char="@", header=FALSE)  
  file <- paste(x, "-5-", i, "tst.dat", sep="")  
  x_tst <- read.csv(file, comment.char="@", header=FALSE)  
  In <- length(names(x_tra)) - 1  
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")  
  names(x_tra)[In+1] <- "Y"  
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")  
  names(x_tst)[In+1] <- "Y"  
  if (tt == "train") {  
    test <- x_tra  
  }  
  else {  
    test <- x_tst  
  }  
  fitMulti=lm(Y~.,x_tra)  
  yprime=predict(fitMulti,test)  
  sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE  
}  
lmMSEtrain2<-mean(sapply(1:5,run_lm_fold,nombre,"train"))  
lmMSEtest2<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
```

A lo que se obtienen los resultados:

```
# Mostramos resultados  
print('Resultados modelo lineal con todas las variables')
```

```
## [1] "Resultados modelo lineal con todas las variables"  
print(lmMSEtrain2)
```

```
## [1] 4.81969
```

```
print(lmMSEtest2)
```

```
## [1] 4.942255
```

Con un error cuadrático medio de 4.81969 para la media de los subconjuntos de entrenamiento, y 4.942255 para los de evaluación.

Para la evaluación de los algoritmos de regresión, lo primero que debemos hacer es cargar las tablas proporcionadas con los resultados para distintos datasets.

```
#leemos la tabla con los errores medios de test
resultados <- read.csv("input/Tablas/regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]

#leemos la tabla con los errores medios de entrenamiento
resultados <- read.csv("input/Tablas/regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]
```

Sustituimos los valores para nuestro dataset con los resultados obtenidos de ambos experimentos.

```
# sustituimos valores obtenidos
# en train
tablatra['abalone','out_train_lm'] <- lmMSEtrain2
tablatra['abalone','out_train_kknn'] <- knnMSEtrain
# en test
tablatst['abalone','out_test_lm'] <- lmMSEtest2
tablatst['abalone','out_test_kknn'] <- knnMSEtest
```

El siguiente paso es normalizar la tabla de resultados para ambas tablas.

```
# Normalizamos la tabla con el código propuesto
##TABLA NORMALIZADA - lm (other) vs knn (ref) para WILCOXON
#+ 0.1 porque wilcox R falla para valores == 0 en la tabla

# train
difs <- (tablatra[,1] - tablatra[,2]) / tablatra[,1]
wilc_1_2.tra <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0,     abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2.tra) <- c(colnames(tablatra)[1], colnames(tablatra)[2])
head(wilc_1_2.tra)

##      out_train_lm out_train_kknn
## [1,]      0.1    0.6398543
## [2,]      0.1    1.0629412
## [3,]      0.1    0.7873339
## [4,]      0.1    0.7709917
## [5,]      0.1    0.6490708
## [6,]      0.1    0.7765836

# Test
difs <- (tablatst[,1] - tablatst[,2]) / tablatst[,1]
wilc_1_2.tst <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0,     abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2.tst) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
head(wilc_1_2.tst)

##      out_test_lm out_test_kknn
## [1,]  0.1922483   0.1000000
## [2,]  0.1000000   1.0294118
## [3,]  0.1000000   0.4339071
```

```

## [4,] 0.1000000 0.3885965
## [5,] 0.1548506 0.1000000
## [6,] 0.1000000 0.3061057

```

Aplicamos el test de Wilcoxon para comparar el modelo de regresión múltiple con k-nn. Primero lo haremos sobre el subconjunto de entrenamiento.

```

#Aplicación del test de WILCOXON
# subconjunto de train
LMvsKNNtra <- wilcox.test(wilc_1_2.traj[,1], wilc_1_2.traj[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtra$statistic
pvalue <- LMvsKNNtra$p.value
LMvsKNNtra <- wilcox.test(wilc_1_2.traj[,2], wilc_1_2.traj[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtra$statistic
Rmas

## V
## 10
Rmenos

## V
## 161
pvalue

## [1] 0.000328064

```

Con un p-valor de 0.7660294 no podemos afirmar de que haya diferencias estadísticamente significativas entre ambos algoritmos.

Para el subconjunto de “test”:

```

# subconjunto de test
LMvsKNNtst <- wilcox.test(wilc_1_2.tst[,1], wilc_1_2.tst[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2.tst[,2], wilc_1_2.tst[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic
Rmas

## V
## 78
Rmenos

## V
## 93
pvalue

## [1] 0.7660294

```

Se obtiene el mismo resultado que en el subconjunto de entrenamiento. Por lo que no podemos afirmar que haya diferencias estadísticamente significativas entre ambos algoritmos.

El último paso sería comparar estos dos algoritmos junto al algoritmo M5' (cuyos resultados tenemos en las tablas) aplicando el test de Friedman. Se aplicará a ambos subconjuntos.

```

#Aplicación del test de Friedman
# Para train
test_friedman.tra <- friedman.test(as.matrix(tablatra))
test_friedman.tra

```

```

##  

## Friedman rank sum test  

##  

## data: as.matrix(tablatra)  

## Friedman chi-squared = 20.333, df = 2, p-value = 3.843e-05  

# para test  

test_friedman.tst <- friedman.test(as.matrix(tablatst))  

test_friedman.tst

```

```

##  

## Friedman rank sum test  

##  

## data: as.matrix(tablatst)  

## Friedman chi-squared = 8.4444, df = 2, p-value = 0.01467

```

Obtenemos para ambas tablas un p-valor pequeño (menor a 0.05), que nos indica de que tenemos suficiente evidencia estadística para considerar que al menos dos de los algoritmos son diferentes entre ellos. Para obtener las comparativas, aplicamos post-hoc de Holm.

```

#Aplicación del test post-hoc de HOLM  

# train  

tam.tra <- dim(tablatra)  

groups.tra <- rep(1:tam.tra[2], each=tam.tra[1])  

pairwise.wilcox.test(as.matrix(tablatra), groups.tra, p.adjust = "holm", paired = TRUE)

```

```

##  

## Pairwise comparisons using Wilcoxon signed rank exact test  

##  

## data: as.matrix(tablatra) and groups.tra  

##  

##    1      2  

## 2 0.0031 -  

## 3 0.0032 0.0032  

##  

## P value adjustment method: holm  

# test  

tam.tst <- dim(tablatst)  

groups.tst <- rep(1:tam.tst[2], each=tam.tst[1])  

pairwise.wilcox.test(as.matrix(tablatst), groups.tst, p.adjust = "holm", paired = TRUE)

```

```

##  

## Pairwise comparisons using Wilcoxon signed rank exact test  

##  

## data: as.matrix(tablatst) and groups.tst  

##  

##    1      2  

## 2 0.580 -  

## 3 0.081 0.108  

##  

## P value adjustment method: holm

```

Para la tabla con los resultados de los entrenamientos obtenemos un p-valor menor a 0.05 para todas las comparaciones, por lo que se puede decir con un 95% de confianza que los tres algoritmos tienen diferencias estadísticamente significativas.

Si nos fijamos en los resultados para la tabla de “test” este resultado es bastante diferente. En este caso obtenemos la tabla con la comparativa entre los tres algoritmos, siendo el que mayor diferencias tiene M5 con un intervalo de confianza cercano al 90%, puesto que los p-valores obtenidos son: 0.081 con respecto a lm y 0.108 con respecto a knn. Mientras knn y lm obtienen un p-valor de 0.580 entre ellos.