

Análisis de Datos: Precio de los coches en la India

Amanda del Álamo & Alejandro Delgado

16/11/2021

Contents

DATASET	1
DEFINICIÓN DE OBJETIVOS	2
ANÁLISIS EXPLORATORIO INICIAL	2
SEPARACION TRAIN/TEST	4
DETECCIÓN, TRATAMIENTO E IMPUTACIÓN DE DATOS	17
TRANSFORMACIÓN Y CREACIÓN DE VARIABLES	20
SELECCIÓN DE VARIABLES	21
AJUSTE, INTERPRETACIÓN Y DIAGNOSIS DEL MODELO DE REGRESIÓN LINEAL MÚLTIPLE	50
PREDICCIÓN SOBRE LOS DATOS DE TESTING. EVALUACIÓN DEL MODELO	92

DATASET

```
data <- read.csv("cochesdataP.csv", na = c("", "NA", "NULL", NULL, " ", "/n" ))
head(data)

##   X                  Name  Location Year Kilometers_Driven
## 1 0      Maruti Wagon R LXI CNG    Mumbai 2010          72000
## 2 1 Hyundai Creta 1.6 CRDi SX Option    Pune 2015          41000
## 3 2            Honda Jazz V    Chennai 2011          46000
## 4 3      Maruti Ertiga VDI    Chennai 2012          87000
## 5 4  Audi A4 New 2.0 TDI Multitronic Coimbatore 2013          40670
## 6 5  Hyundai EON LPG Era Plus Option Hyderabad 2012          75000
##   Fuel_Type Transmission Owner_Type Mileage Engine     Power Seats
## 1       CNG        Manual First 26.6 km/kg 998 CC 58.16 bhp 5
## 2     Diesel        Manual First 19.67 kmpl 1582 CC 126.2 bhp 5
## 3     Petrol        Manual First 18.2 kmpl 1199 CC 88.7 bhp 5
## 4     Diesel        Manual First 20.77 kmpl 1248 CC 88.76 bhp 7
## 5     Diesel      Automatic Second 15.2 kmpl 1968 CC 140.8 bhp 5
## 6       LPG        Manual First 21.1 km/kg 814 CC 55.2 bhp 5
##   New_Price Price
## 1       <NA>  1.75
```

```

## 2      <NA> 12.50
## 3 8.61 Lakh  4.50
## 4      <NA>  6.00
## 5      <NA> 17.74
## 6      <NA>  2.35

```

DEFINICIÓN DE OBJETIVOS

Este dataset contiene toda la información acerca de un comercio de venta de coches usados en la India. Nuestro objetivo es predecir, a través de las características de un coche, cuanto nos costaría comprarlo.

ANÁLISIS EXPLORATORIO INICIAL

La función glimpse() del paquete dplyr puede utilizarse para ver las columnas del conjunto de datos y mostrar alguna porción de ellos con respecto a cada atributo que pueda caber en una sola línea.

```
glimpse(data)
```

```

## #> #> Rows: 6,019
## #> #> Columns: 14
## #> #> $ X          <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15~
## #> #> $ Name        <chr> "Maruti Wagon R LXI CNG", "Hyundai Creta 1.6 CRDi SX~
## #> #> $ Location     <chr> "Mumbai", "Pune", "Chennai", "Chennai", "Coimbatore"~
## #> #> $ Year         <int> 2010, 2015, 2011, 2012, 2013, 2012, 2013, 2016, 2013~
## #> #> $ Kilometers_Driven <int> 72000, 41000, 46000, 87000, 40670, 75000, 86999, 360~
## #> #> $ Fuel_Type     <chr> "CNG", "Diesel", "Petrol", "Diesel", "Diesel", "LPG"~
## #> #> $ Transmission   <chr> "Manual", "Manual", "Manual", "Manual", "Automatic", ~
## #> #> $ Owner_Type     <chr> "First", "First", "First", "First", "Second", "First"~
## #> #> $ Mileage        <chr> "26.6 km/kg", "19.67 kmpl", "18.2 kmpl", "20.77 kmpl"~
## #> #> $ Engine          <chr> "998 CC", "1582 CC", "1199 CC", "1248 CC", "1968 CC"~
## #> #> $ Power           <chr> "58.16 bhp", "126.2 bhp", "88.7 bhp", "88.76 bhp", "~~
## #> #> $ Seats            <dbl> 5, 5, 5, 7, 5, 5, 8, 5, 5, 5, 5, 5, 5, 7, 5, 5, 5~
## #> #> $ New_Price       <chr> NA, NA, "8.61 Lakh", NA, NA, NA, NA, "21 Lakh", NA, ~
## #> #> $ Price            <dbl> 1.75, 12.50, 4.50, 6.00, 17.74, 2.35, 3.50, 17.50, 5~

```

Con esta forma de visualización podemos observar mejor la naturaleza de los datos. Tenemos 6019 observaciones y 14 columnas y a simple vista vemos que muchos datos están bien tipados pero otros habrá que transformarlos, como puede ser el caso de la variable Mileage (Kilometraje) que está como Character ya que viene el número con su unidad de medida y habrá que decidir si ponerlo todo como km/kg o kmpl. Lo mismo nos pasa con Power (potencia) y New Price que parece tener muchos elementos faltantes. La variable Seats (asientos) es un double cuando debería ser entera... Hay muchas cosas que modificar para dar valor a estos datos.

1. Name: Nombre de las marcas y modelos de los coches.
2. Location: La ubicación en la que se vende el coche o está disponible para su compra.
3. Year: El año del modelo de coche.
4. Kilometers_Driven: El total de kilómetros recorridos en el coche por el anterior propietario en Km
5. Fuel_Type: El tipo de combustible utilizado por el coche.
6. Transmission: El tipo de transmisión utilizado por el coche.
7. Owner_Type: Indica si el coche es de primera, segunda, tercera, cuarta... mano.
8. Mileage: El kilometraje estándar ofrecido por la compañía de automóviles en kmpl o km/kg.
9. Engine: La cilindrada del motor en cc.
10. Power: La potencia máxima del motor en bhp (CV).
11. Seats: Número de asientos.
12. New_Price: El precio del coche, nuevo.

13. Price: El precio del coche usado en INR Lakhs (Rupias).

Con la función anterior hemos podido apreciar que hay datos que contienen diferentes tipos de unidades. Vamos a ver cuales son y como se dividen antes de limpiar los datos para luego saber como podemos tratarlas.

Relación entre unidades

Mileage

```
data %>% pull(Mileage) %>%
  str_split(" ", simplify=TRUE) %>%
  cbind(data$Fuel_Type) %>%
  subset(select=2:3) %>%
  as.data.frame() %>% table() #Extraigo los diferentes tipos que hay y como se relacionan

##          V2
## V1      CNG Diesel Electric LPG Petrol
##      0     0      2    0    0
## km/kg  56     0      0   10    0
## kmp1   0    3205      0    0  2746
```

Engine

```
data %>% pull(Engine) %>%
  str_split(" ", simplify=TRUE) %>%
  subset(select=2) %>% table() # hago lo mismo para confirmar que solo hay un tipo

## .
##      CC
## 36 5983
```

Power

```
data %>% pull(Power) %>%
  str_split(" ", simplify=TRUE) %>%
  subset(select=2) %>% table() # hago lo mismo para confirmar que solo hay un tipo

## .
##      bhp
## 36 5983
```

New Price

```
data %>% pull(New_Price) %>%
  str_split(" ", simplify=TRUE) %>%
  subset(select=2) %>% table()

## .
##      Cr Lakh
## 5195 17 807
```

Podemos observar que en Mileage y New_Price tenemos diferentes tipos de unidades. Las diferentes unidades en Mileage tienen que ver con el tipo de combustible y las de New_Price se dividen en Rupias (Lakh) y

Colón (Cr) Más adelante dividiremos en dos variables distintas.

```
data %>% mutate(Mileage = as.numeric(str_extract(Mileage, "^[[:graph:]+"))) -> data
data %>% mutate(Engine = as.numeric(str_extract(Engine, "^[[:graph:]+"))) -> data
data %>% mutate(Power = as.numeric(str_extract(Power, "^[[:graph:]+"))) -> data
data %>%
  mutate(Seats = as.integer(Seats)) -> data
data %>%
  mutate(Fuel_Type = as.factor(Fuel_Type)) %>%
  mutate(Transmission = as.factor(Transmission)) %>%
  mutate(Location = as.factor(Location)) %>%
  mutate(New_Price = as.numeric(str_extract(New_Price, "^[[:graph:]+"))) -> data
data %>% mutate(Owner_Type = factor(Owner_Type, levels=c("First", "Second", "Third", "Fourth & Above")))
```

Hemos cambiado el tipo de variable de algunas de las columnas ya que no era el adecuado. A la variable Power se le introducen NA's por coerción debido a que tiene algun valor recogido como NA. También hemos modificado la grañía del dato ya que venían con las unidades en el registro.

SEPARACION TRAIN/TEST

Procedemos a separar el dataset en sets de train y de test

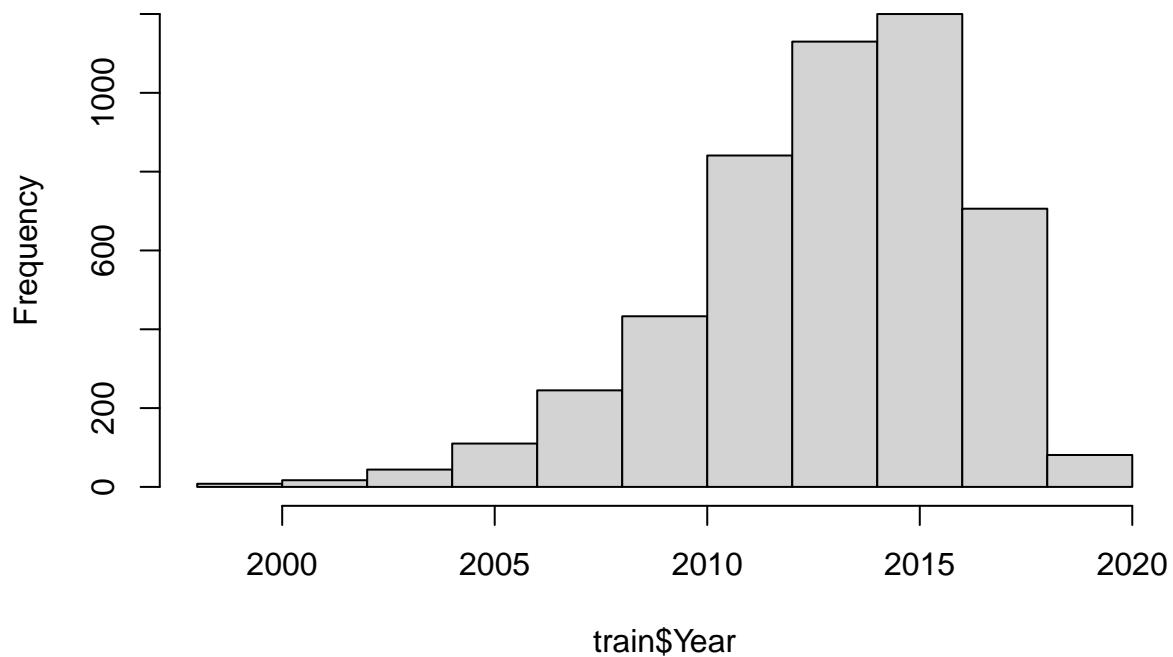
```
# 80% del tamaño de la muestra
smp_size <- floor(0.8 * nrow(data))
# establecemos la semilla para que su partición sea reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)
train <- data[train_ind, ]
test <- data[-train_ind, ]
rm(train_ind)
```

Análisis variable “Year”

Histograma

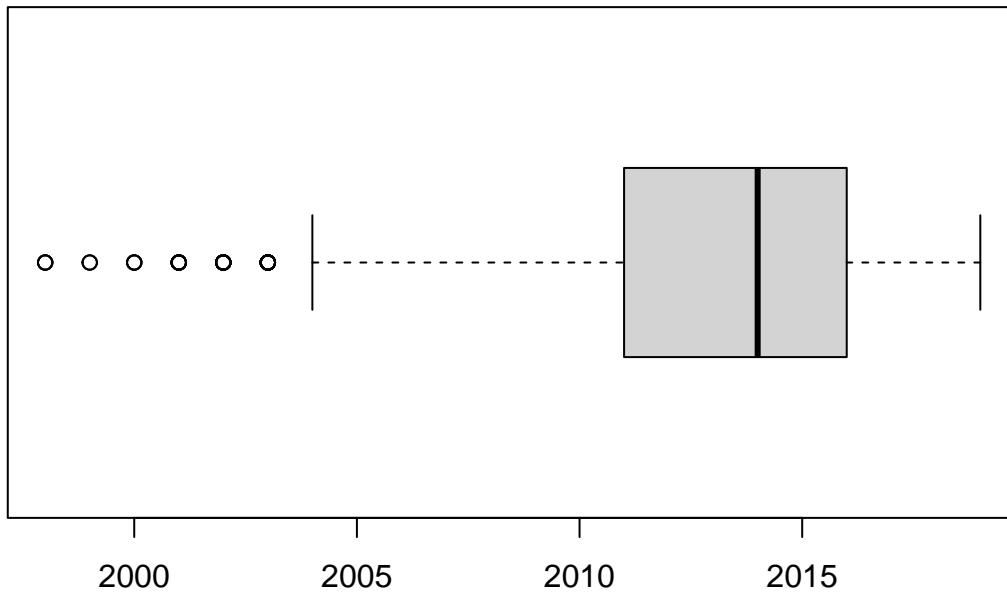
```
hist(train$Year)
```

Histogram of train\$Year



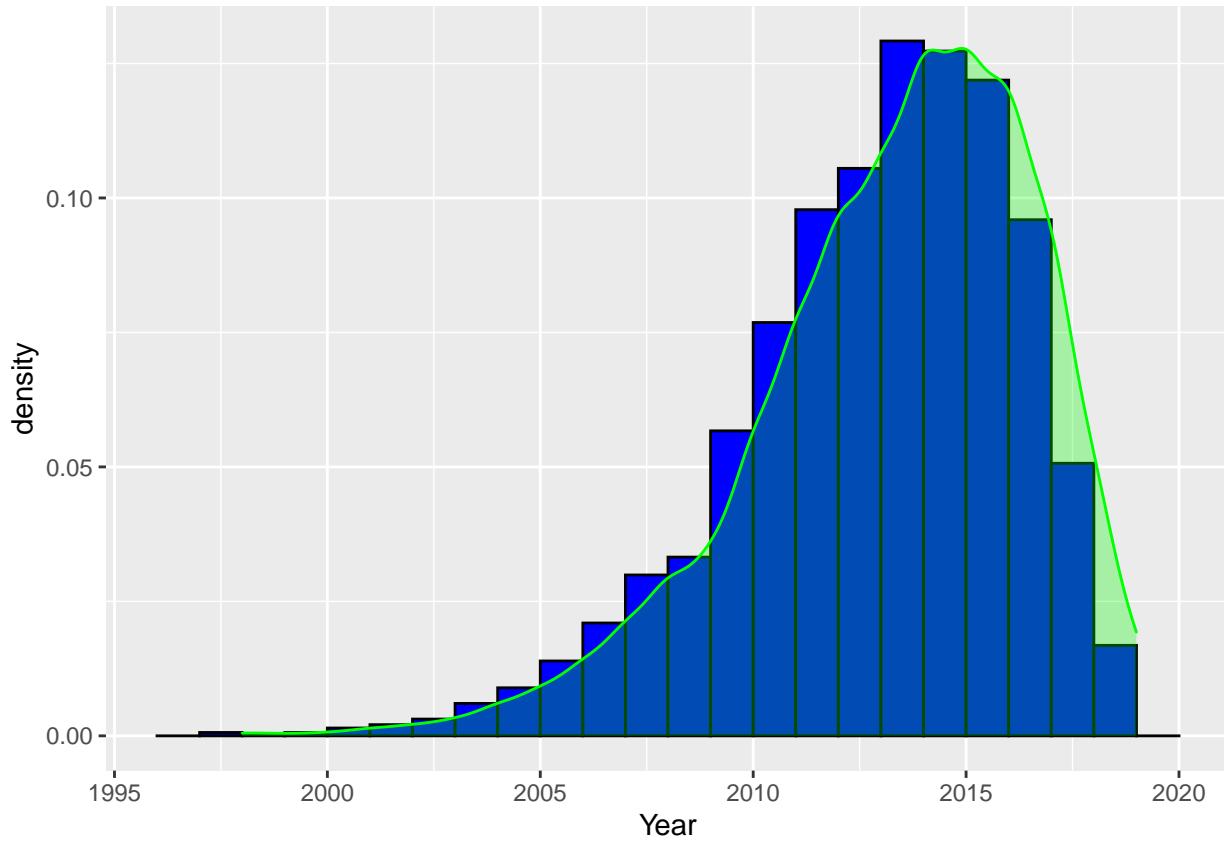
Boxplot

```
boxplot(train$Year, horizontal = TRUE)
```



Histograma + Densidad

```
ggplot(train, aes(x=Year)) +  
  geom_histogram(breaks=seq(1996,2020,by=1),aes(y=..density..), binwidth = 0.4, fill="blue", colour="blue")  
  geom_density(colour="green", fill="green",alpha=0.3)
```



La primera variable que vamos a analizar es “Year”, hace referencia al año en que se fabricó cada coche. Se trata de una variable cuantitativa, ya que podemos calcular la diferencia entre fechas, en la que aparecen los años desde 1996 hasta 2018. Obtenemos que en el año 1998 se contabilizan 10 coches, en el año 2000, 23, en el 2004, 135. Va aumentando el número de coches progresivamente con el paso de los años hasta llegar a 2016 donde vuelve a disminuir también progresivamente. Esta variable está definida como “integer” y así la trataremos, ya que no nos da problema y nos aporta toda la información necesaria. Además, gracias al boxplot se aprecia que el 50% de los datos se encontrarán entre 2011 y 2017.

Análisis variable “Kilometers_driven”

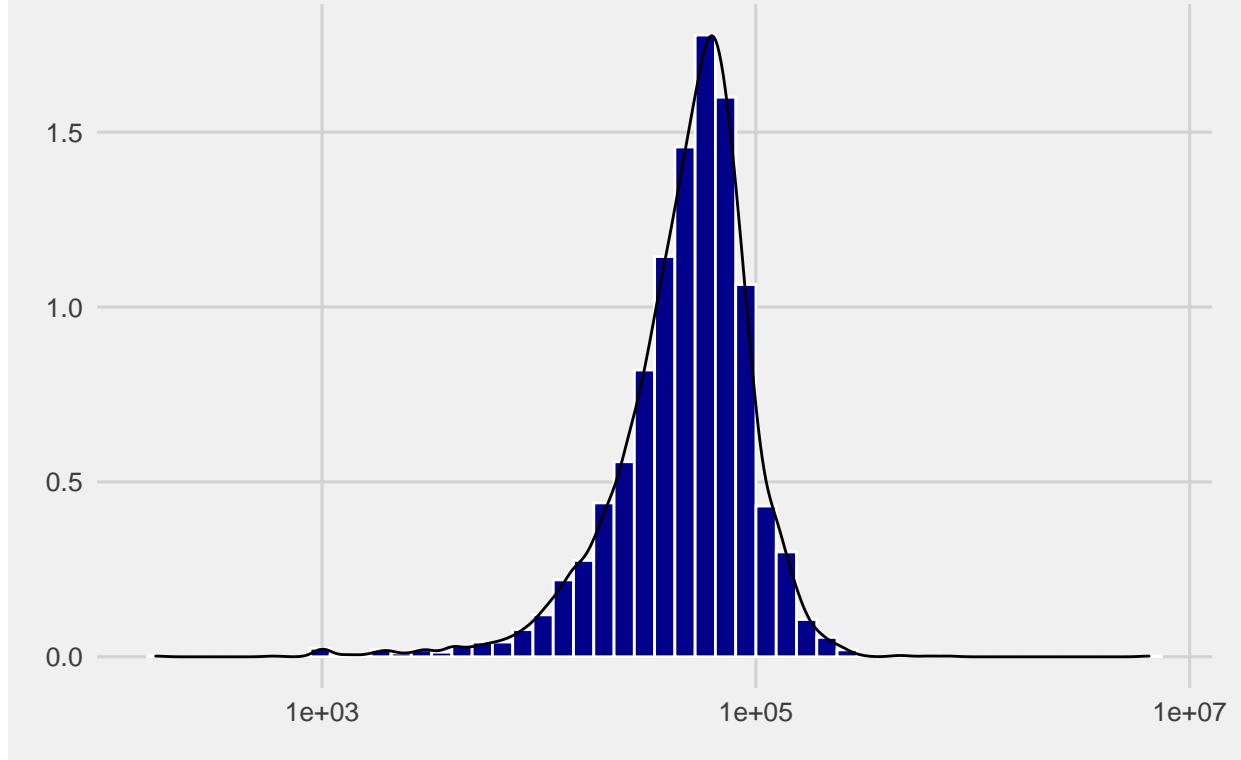
Summary

```
summary(train$Kilometers_Driven)
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##     171    33900   52782    59074   73194 6500000
```

Histograma + Densidad

```
train %>% ggplot(aes(x=Kilometers_Driven)) +
  ggtitle("Kilómetros recorridos") +
  theme_fivethirtyeight() +
  geom_histogram(aes(y=..density..), color="white", fill="dark blue", bins = 50) +
  geom_density() +
  scale_x_log10()
```

Kilómetros recorridos



La segunda variable a analizar es “Kilometers_driven”, que hace referencia a los kilómetros que lleva recorridos el coche. Se trata de una variable cuantitativa expresada como integer en nuestros datos. El mínimo de kilómetros recorridos por un coche de los que estamos estudiando es de 171 mientras que el máximo llega hasta 6500.000. Analizándolo al detalle se aprecia que el valor máximo es un outlier ya que está muy lejos de la media y es el único coche con tantos km recorridos. Más adelante, decidiremos qué hacer con ese valor, de momento, obtenemos que la media de kilómetros recorridos por los coches es de 58738 y que el 50% de los datos se encuentran entre 53000 y 73000. Además hemos aplicado escala logarítmica en el histograma para una mejor visualización.

```
train %>% filter(Kilometers_Driven>4e5)
```

```
##      X                               Name Location Year Kilometers_Driven
## 1 2328          BMW X5 xDrive 30d M Sport    Chennai 2017       6500000
## 2 4491        Hyundai i20 Magna Optional 1.2 Bangalore 2013       445000
## 3 340 Skoda Octavia Ambition Plus 2.0 TDI AT   Kolkata 2013       775000
## 4 3092           Honda City i VTEC SV   Kolkata 2015       480000
## 5 358        Hyundai i10 Magna 1.2    Chennai 2009       620000
##   Fuel_Type Transmission Owner_Type Mileage Engine Power Seats New_Price Price
## 1 Diesel     Automatic    First    15.97  2993 258.0     5      NA  65.00
## 2 Petrol     Manual     First    18.50  1197  82.9     5      NA  4.45
## 3 Diesel     Automatic    First    19.30  1968 141.0     5      NA  7.50
## 4 Petrol     Manual     First    17.40  1497 117.3     5      NA  5.00
## 5 Petrol     Manual     First    20.36  1197  78.9     5      NA  2.70
```

Es llamativo que un coche del año 2017 tenga 6500000 kilómetros recorridos, pero puede ser posible, así que

lo dejamos así.

Análisis variable “Fuel_Type”

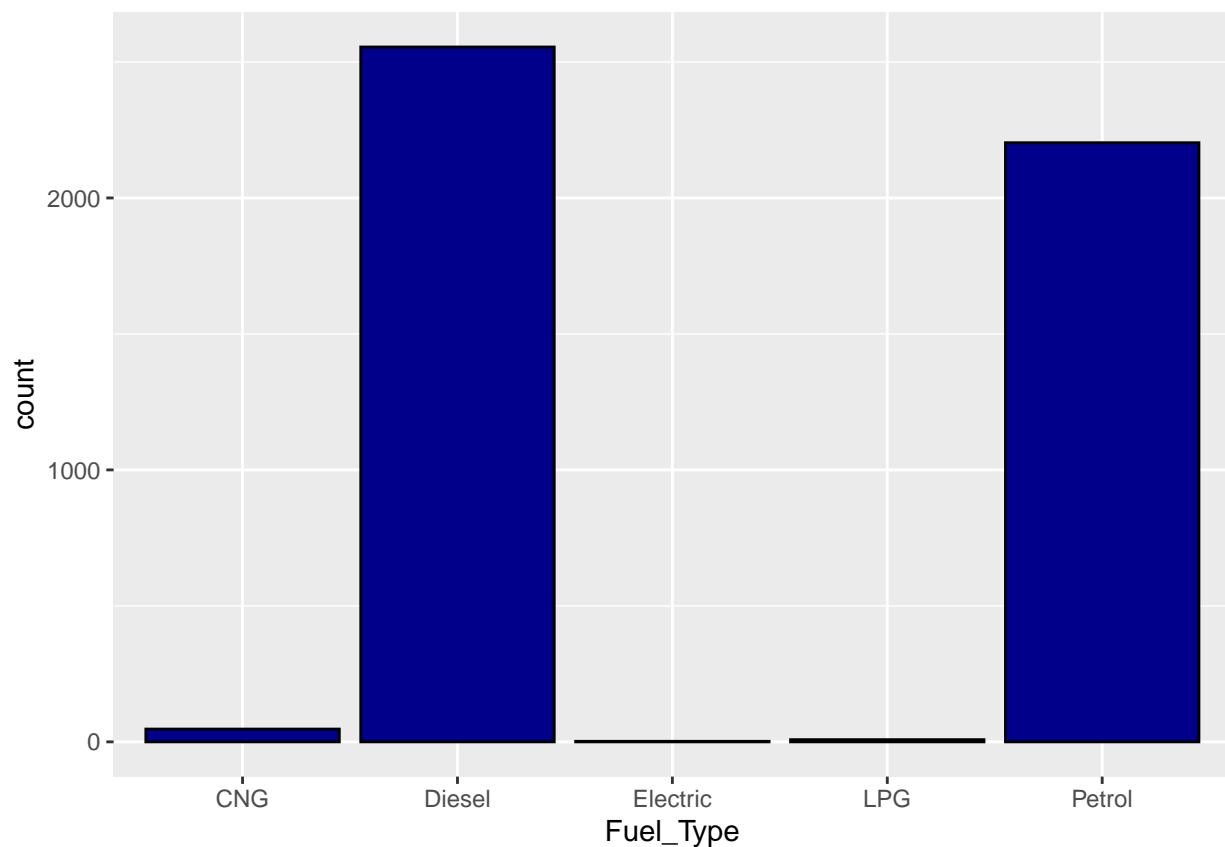
Summary

```
summary(train$Fuel_Type)
```

```
##      CNG    Diesel Electric       LPG    Petrol
##      47     2555        2       8    2203
```

Histograma

```
ggplot(train, aes(x=Fuel_Type)) +
  geom_histogram(stat="count", binwidth = 0.4, fill="dark blue", colour="black", )
```



La tercera variable que vamos a analizar es “Fuel_Type”, que hace referencia al tipo de combustible que usa cada coche. Se trata de una variable cualitativa. Está representada en nuestros datos como factor y los valores que toma son CNG,LPG, Electric, Diesel y Pretol. Vamos a describir cada uno de los valores que toma esta variable:

-CNG: El gas natural comprimido, más conocido por la sigla GNC, es un combustible para uso vehicular que, por ser económico y ambientalmente más limpio, es considerado una alternativa sustentable para la sustitución de combustibles líquidos.

-LPG: El gas licuado del petróleo es la mezcla de gases licuados presentes en el gas natural o disueltos en el petróleo. Lleva consigo procesos físicos y químicos por ejemplo el uso de metano.

-Diésel: El diésel o dísel, también denominado gasóleo o gasoil, es un hidrocarburo líquido de densidad sobre 850 kg/m³, compuesto fundamentalmente por parafinas y utilizado principalmente como combustible en calefacción y en motores diésel. Su poder calorífico inferior es de 35,86 MJ/l.

-Petrol: La gasolina es formada con el petróleo refinado, utilizado principalmente como combustible, es esencial para la red mundial de transporte, el combustible primario que hace funcionar los motores de combustión interna que mueven la mayoría de los automóviles y otros sistemas de transporte.

De Electric, eléctricos es del tipo que menos muestras poseemos, sólo 2. De LPG y CNG también poseemos pocas (1 y 56 respectivamente) y es en Diesel y Petrol donde tenemos la mayoría de los datos, siendo 3852 y 3325 las muestras respectivamente. Se muestra el histograma para visualizar la gran diferencia de muestras entre Diesel y Petróleo y las demás.

Análisis variable “Location”

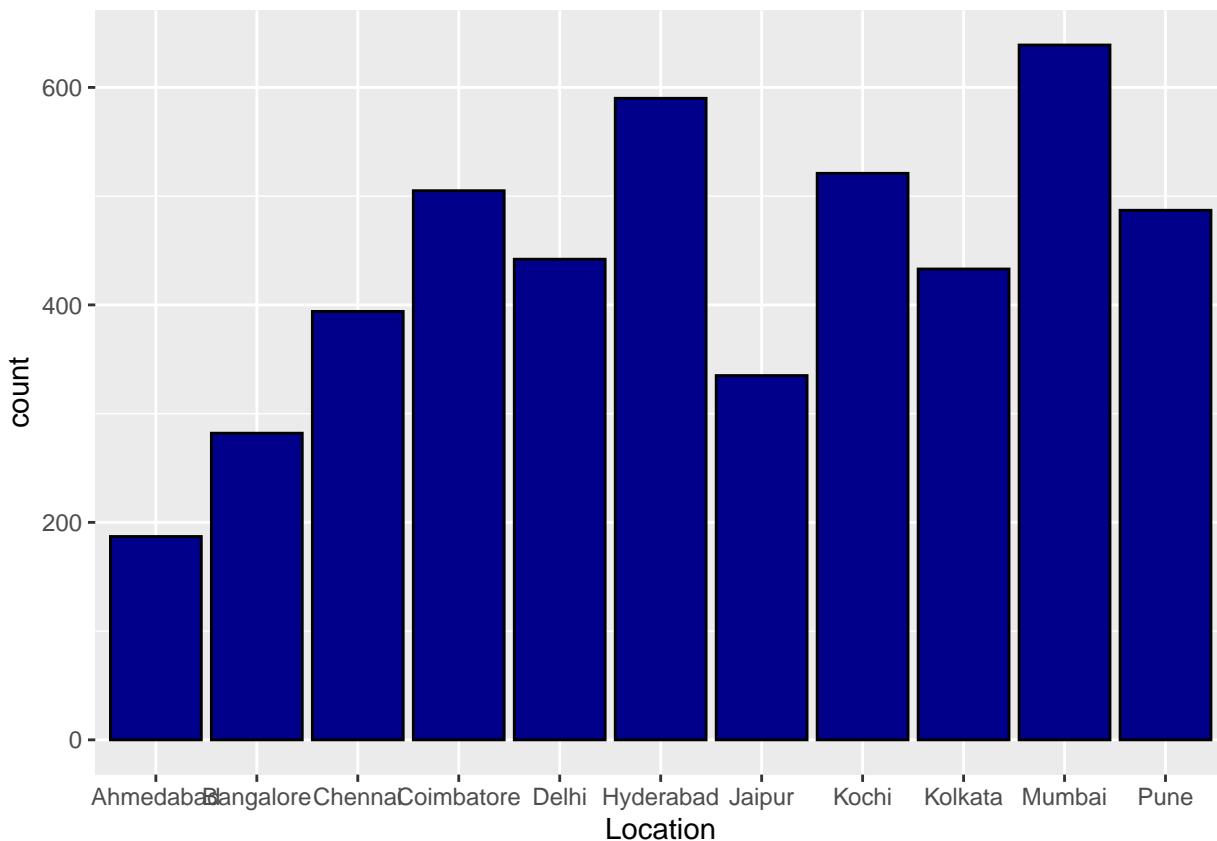
Summary

```
summary(train$Location)
```

```
##   Ahmedabad   Bangalore    Chennai Coimbatore      Delhi Hyderabad      Jaipur
##       187          282        394         505       442         590       335
##      Kochi        Kolkata     Mumbai      Pune
##       521          433        639         487
```

Histograma

```
ggplot(train, aes(x=Location)) +
  geom_histogram(stat="count", binwidth = 0.4, fill="dark blue", colour="black")
```



Otra de las variables es Location, que es la localización en que está disponible el coche para vender. Se trata de una variable cuantitativa que catalogamos como factor para estudiarla en función de las muestras que posea cada ciudad. La que menos coches disponibles tiene por el momento es Ahmedabad con 224 y le secunda Bangalore con 358 y por el otro extremo, tenemos a la ciudad de Hyderabad con 742 y Mumbai con el máximo de coches disponibles, 790.

Análisis variable “Transmission”

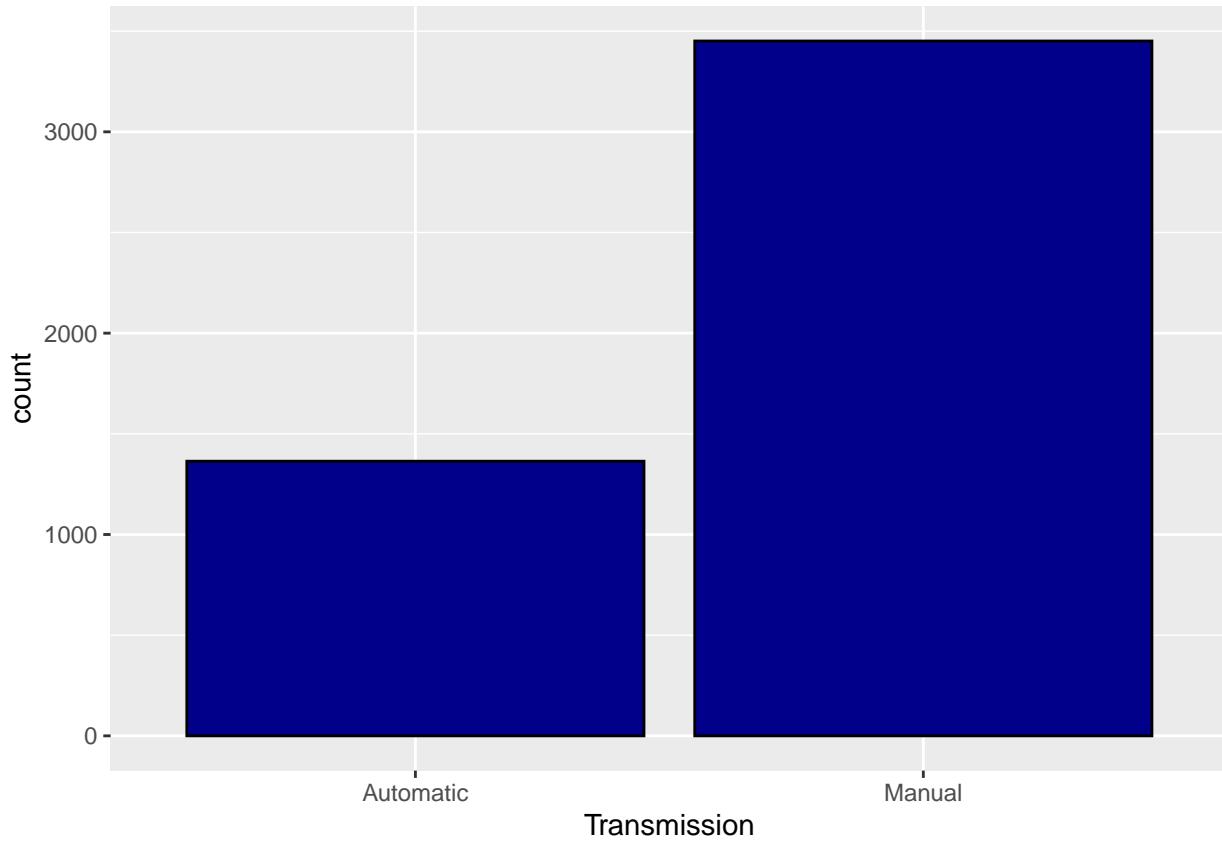
Summary

```
summary(train$Transmission)
```

```
## Automatic      Manual
##        1364      3451
```

Histograma

```
ggplot(train, aes(x=Transmission)) +
  geom_histogram(stat="count", binwidth = 0.4, fill="dark blue", colour="black", ordered=TRUE)
```



Seguimos con el análisis univariante, ahora con “Transmission”. Se trata de una variable cualitativa binaria, en la que podemos distinguir entre los coches automáticos y manuales. Una transmisión manual es una caja de cambios que no puede alterar la relación de cambio por sí sola, requiriendo la intervención del conductor para hacer esto y sin embargo, una transmisión automática es una caja de cambios de automóviles u otro tipo de vehículos que puede encargarse por sí misma de cambiar la relación de cambio automáticamente a medida que el vehículo se mueve, liberando así al conductor de la tarea de cambiar de marcha manualmente. Se tienen muestras 1720 de automáticos y 4299 de manuales. Tenemos la variable como factor en nuestros datos, en este caso, al ser sólo dos valores, es perfecto para nuestro análisis.

Análisis variable “Owner_Type”

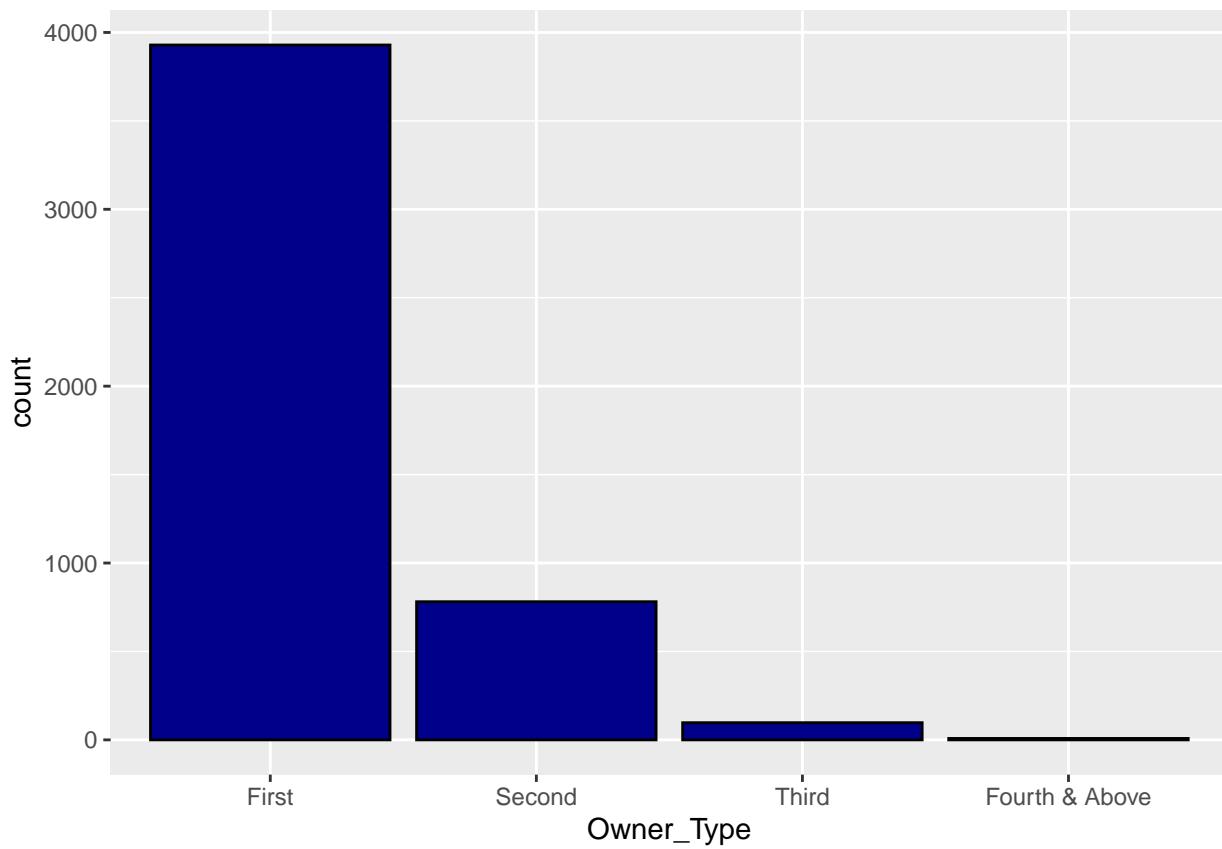
Summary

```
summary(train$Owner_Type)
```

	First	Second	Third	Fourth & Above
##	3929	781	97	8

Histograma

```
ggplot(train, aes(x=Owner_Type)) +
  geom_histogram(stat="count", binwidth = 0.4, fill="dark blue", colour="black")
```



La siguiente variable a analizar es “Owner_Type”, que hace referencia a la clase de propietario, es decir, si el conductor está configurado como primer, segundo o tercero. Se trata de una variable cualitativa en la que podemos distinguir entre los valores: First, Second, Third y Fourth&Above. Del valor que menos muestras tenemos es Fourth&Above con 9 de ellas, después, Third con 113, y en Second y First aumentan considerablemente (968 y 4929, respectivamente) por lo que será mucho más preciso el análisis a la hora de correlacionar variables. Realizamos el histograma para apreciar la diferencia entre el número de muestras de cada tipo.

Análisis variable “Mileage”

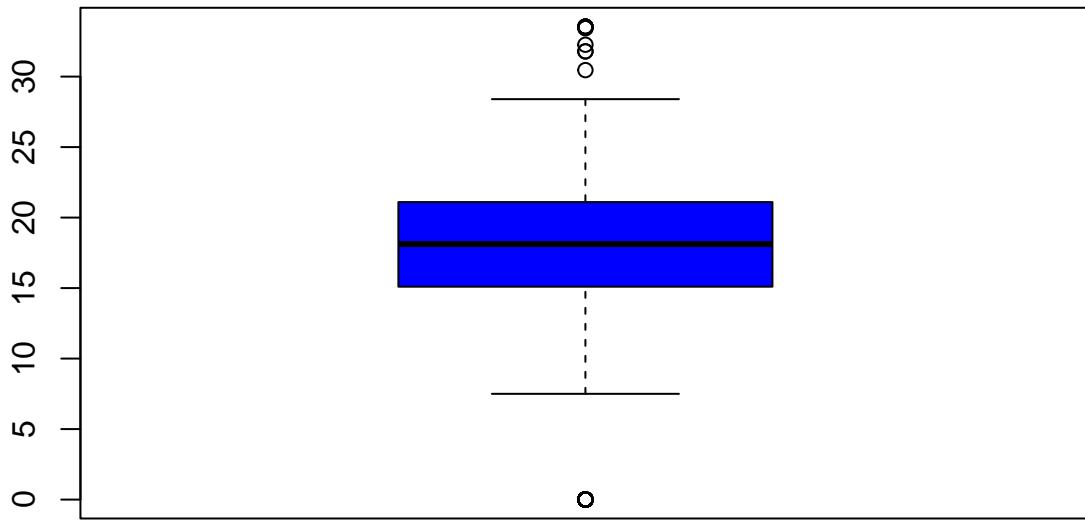
Summary

```
summary(train$Mileage)

##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.    NA's
##      0.00   15.10   18.12   18.12   21.10   33.54       2
```

Boxplot

```
boxplot(train$Mileage, col = "blue")
```



Seguimos con el análisis univariante de la variable “Mileage” que hace referencia al kilometraje del coche y está expresado en kilómetros por litro. Se trata de una variable cuantitativa continua que toma muchos y distintos valores que posteriormente agruparemos por intervalo. En nuestros datos está expresada como factor, lo mejor sería convertirlo a integer o float para así poder realizar gráficos para visualizar cada una de las muestras. No tiene sentido trabajarla como factor pero la trabajaremos más adelante. Tampoco se ha realizado el gráfico porque visualmente no ayuda.

Análisis variable “Engine”

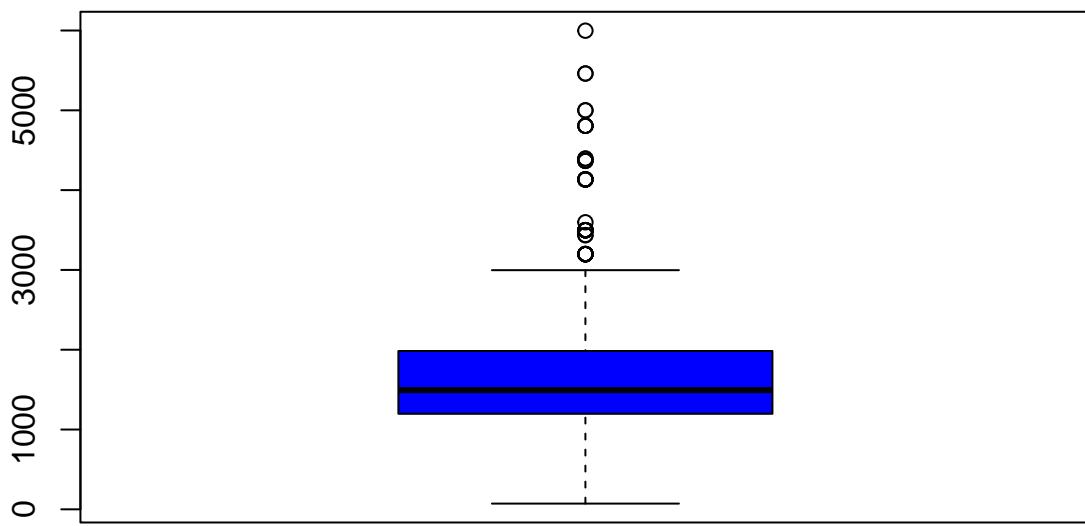
Summary

```
summary(train$Engine)

##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.    NA's
##        72     1197    1493    1620    1984   5998      31
```

Boxplot

```
boxplot(train$Engine, col = "blue")
```



Esta variable, “Engine”, hace referencia a la cilindrada del motor. Se trata de una variable cuantitativa discreta y que está como integer en nuestros datos. El mínimo es 72 y el máximo es 5998 siendo la media 1621 y donde el 50% de las muestras las encontramos entre 1198 y 1984.

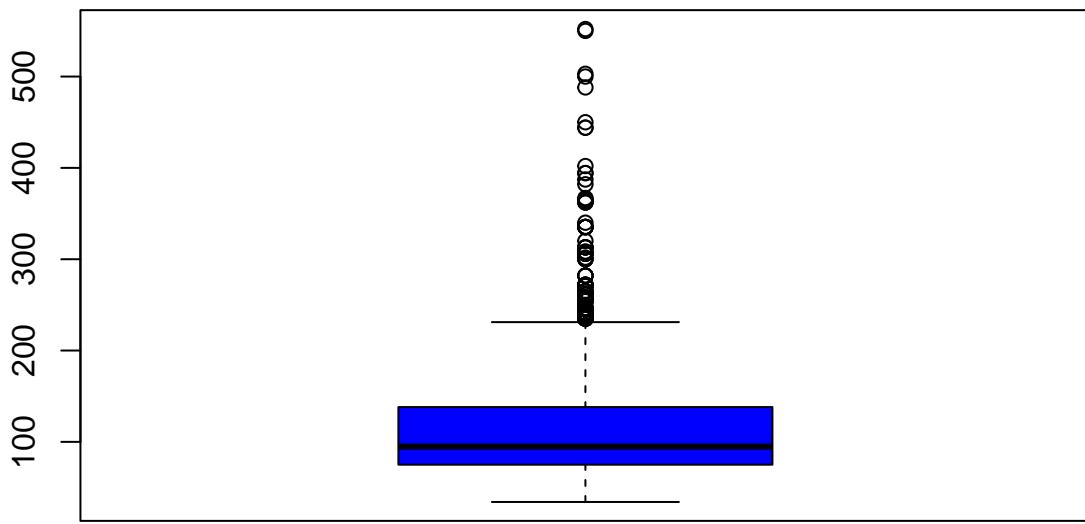
Análisis variable “Power”

Summary

```
summary(train$Power)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     34.20    75.00   94.84  112.96  138.10  552.00     113
```

Boxplot

```
boxplot(train$Power, col = "blue")
```



Esta variable hace referencia a la potencia del motor. Se trata de una variable cuantitativa continua en la que el mínimo es 34.2 y el máximo 560 con media 113.3 y donde el 50% de los datos se encuentran entre 75 y 138. Además, existen 113 valores faltantes que serán de gran importancia en nuestro análisis.

Análisis variable “Seats”

Summary

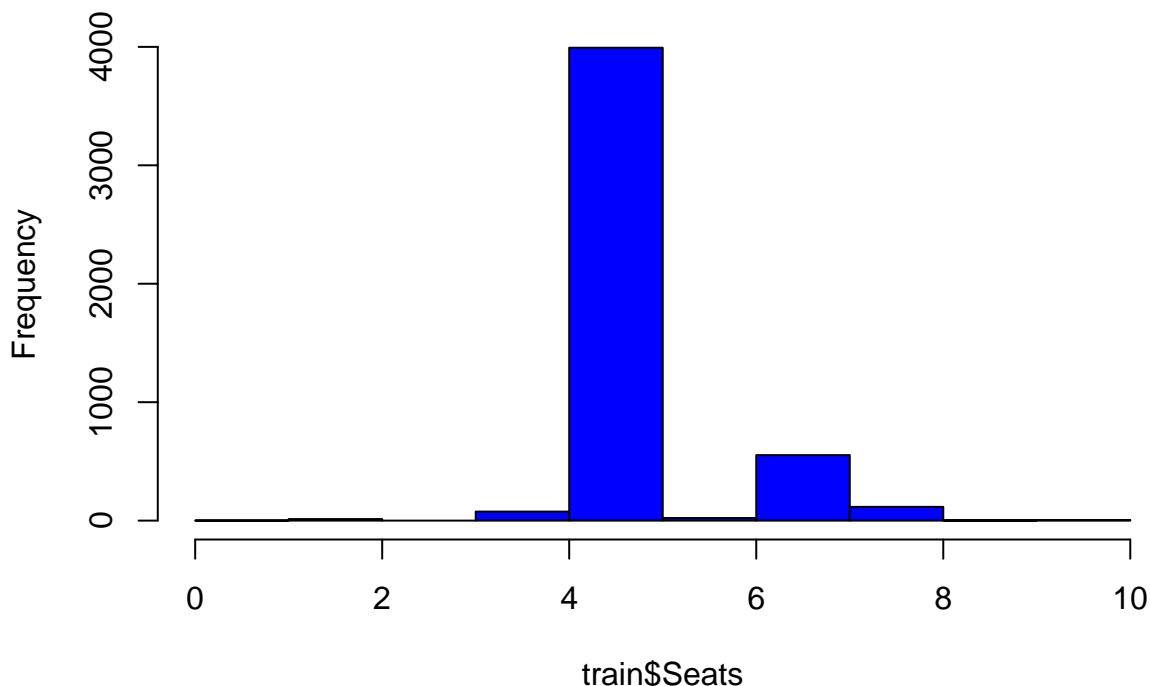
```
summary(train$Seats)

##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.    NA's
##      0.000  5.000  5.000   5.291  5.000 10.000      34
```

Histograma

```
hist(train$Seats, col = "blue")
```

Histogram of train\$Seats



Asientos = 0

No tiene sentido que haya coches con 0 asientos, procedemos a su eliminación

```
train %>% filter(Seats > 0) -> train
summary(train$Seats)
```

```
##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.
##      2.000  5.000  5.000   5.292  5.000 10.000
```

Esta variable “Seats” hace referencia al número de asientos que tiene el coche. Es una variable cuantitativa discreta ya que son valores exactos y además se trata como numeric, que viene perfecto para nuestro análisis. Se aprecia que aparece una de las opciones con Seats=0 cuyo valor es NA, la hemos obviado ya que es imposible que un coche no tenga asientos. La mayoría de los coches poseen 5 asientos y se le acercan también los coches de 7 y 8 plazas, mucho más comunes en personas con muchos hijos o que necesiten los coches para transportar más personas de lo “común”. Si nos fijamos en el summary como hemos dicho antes la media está en 5.28, aproximadamente 5 y contamos con 34 valores NA’s.

Análisis variable “New_Price”

Summary

```
summary(train$New_Price)

##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.    NA's
##      1.00    7.85   11.44    20.31   24.01   95.38    4105
```

NA's %

```
apply(is.na(train[, c(12,13)]), 2, mean) #porcentaje na por columna

##      Seats New_Price
## 0.0000000 0.8587866
```

New_Price contiene 4105 NA's que conforman el 86% de los valores faltantes en esa columna. Más adelante la eliminaremos del train set cuando hagamos la división.

Análisis variable “Price”

Summary

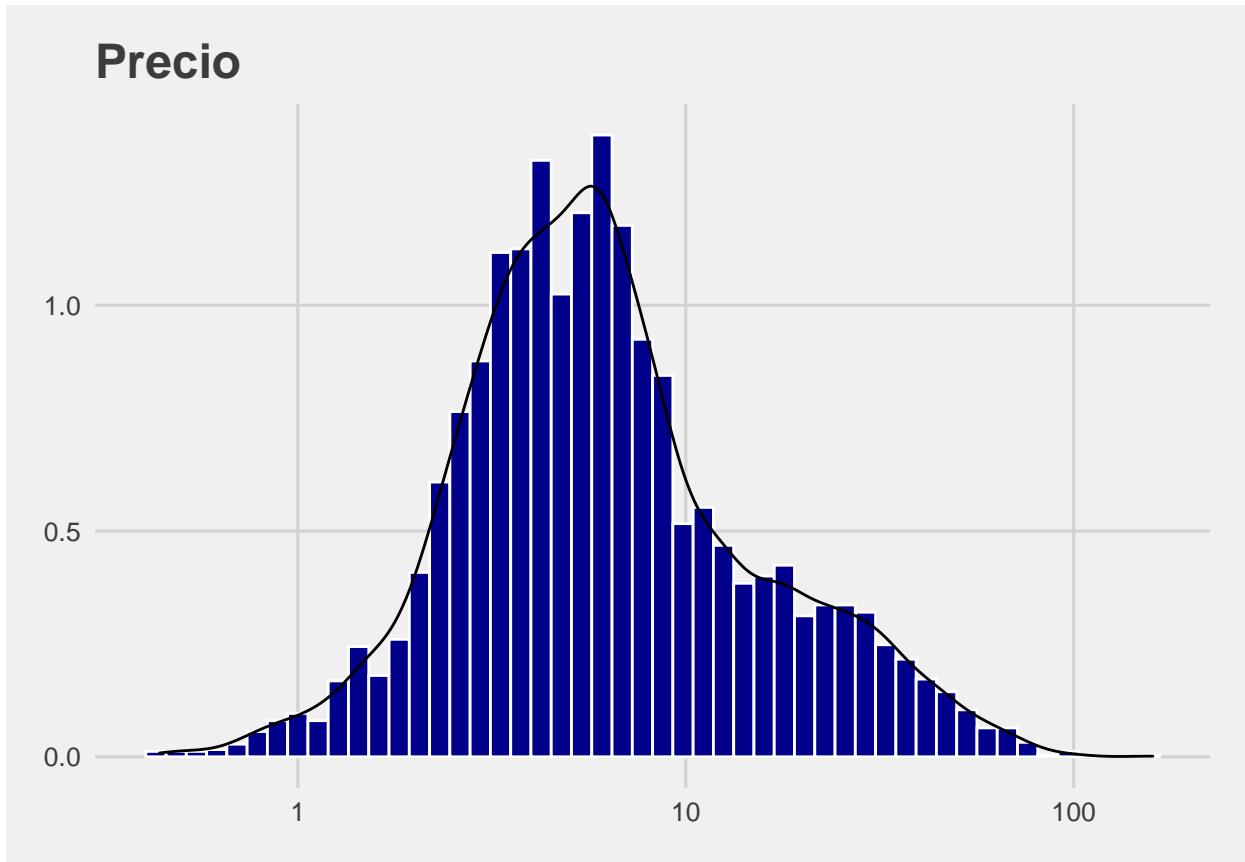
```
summary(train$Price)

##      Min. 1st Qu. Median     Mean 3rd Qu.    Max.
##      0.440    3.500   5.650    9.465   9.928 160.000
```

Histograma + Densidad

Visualizamos Price en escala logarítmica.

```
train%>% ggplot(aes(x=Price)) +
  ggtitle("Precio") +
  theme_fivethirtyeight() +
  geom_histogram(aes(y=..density..), color="white", fill="dark blue", bins = 50) +
  geom_density() +
  scale_x_log10()
```



La variable Price será nuestra variable respuesta. Trataremos de estimar el precio de un coche en base a todas las demás características. El precio está expresado en Rupias y en nuestros datos como numeric. Haciendo un summary vemos que el mínimo precio es de 0.440 y el máximo de 160 que al estar tan lejos de la media se entiende que hay pocos coches que tengan un precio tan alto. La media está en 9.47 rupias y el 50% de los coches tienen un precio de entre 3.5 y 9.92.

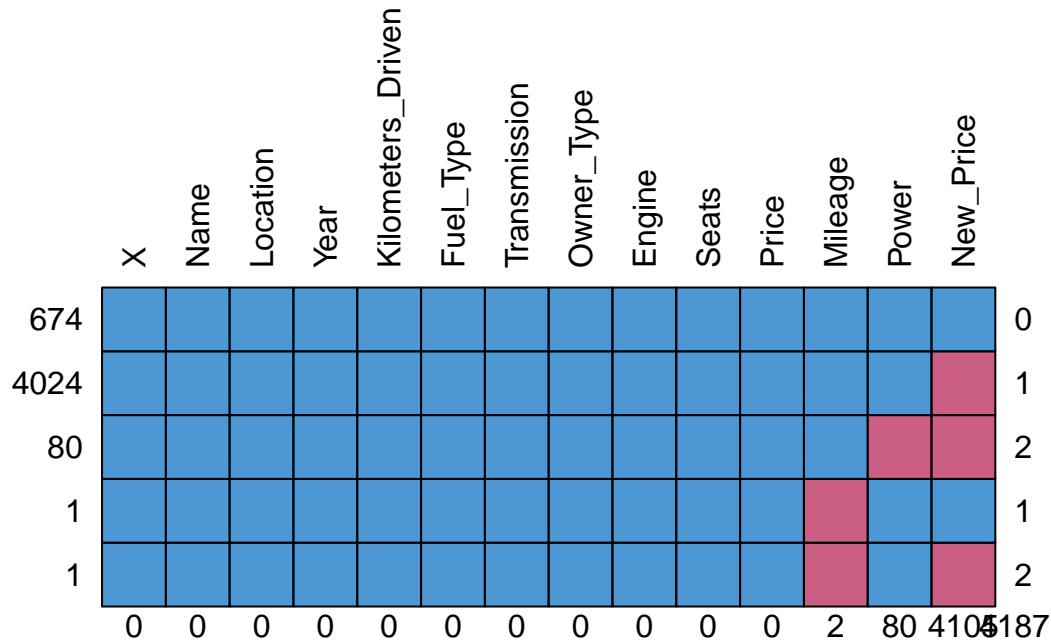
Una vez analizadas tanto las variables de manera univariante como sus outliers, ahora ya podemos detectar y tratar los valores faltantes.

DETECCIÓN, TRATAMIENTO E IMPUTACIÓN DE DATOS

Tabla NA's

A continuación vamos a ver la cantidad de valores perdidos con la función `md.pattern` de la librería `mice`

```
md.pattern(train, plot = TRUE, rotate.names=TRUE)
```



NA's por columna

Vamos a buscar en qué variables y en qué medida de nuestro conjunto de datos hay algún dato del tipo NA

Porcentaje de NA's por columna

```
apply(is.na(train), 2, mean) #porcentaje NA por columna
```

```
##          X           Name        Location       Year
## 0.000000000 0.000000000 0.000000000 0.000000000
## Kilometers_Driven Fuel_Type   Transmission Owner_Type
## 0.000000000 0.000000000 0.000000000 0.000000000
##      Mileage     Engine     Power       Seats
## 0.00041841 0.000000000 0.01673640 0.000000000
##      New_Price    Price
## 0.85878661 0.000000000
```

```
#apply(is.na(train), 2, which) # Posición de NA por columna
#p= dim(train)[2]
#number_na = apply(train[,c(p-1,p)],1,function(x) sum(is.na(x)))
#table(number_na)
```

Número de NA's por columna

```
colSums(is.na(train))
```

```
##          X           Name        Location       Year
## 0           0           0            0            0
## Kilometers_Driven Fuel_Type   Transmission Owner_Type
## 0           0           0            0            0
##      Mileage     Engine     Power       Seats
## 2           0           0            80           0
##      New_Price    Price
## 4105         0           0
```

Se observa claramente que al 86% de los registros les falta datos de la variable New_price, por lo que vamos a proceder a su eliminación ya que la imputación de NA es inviable. También eliminaremos la columna 'X'

ya que no es más que el indice duplicado.

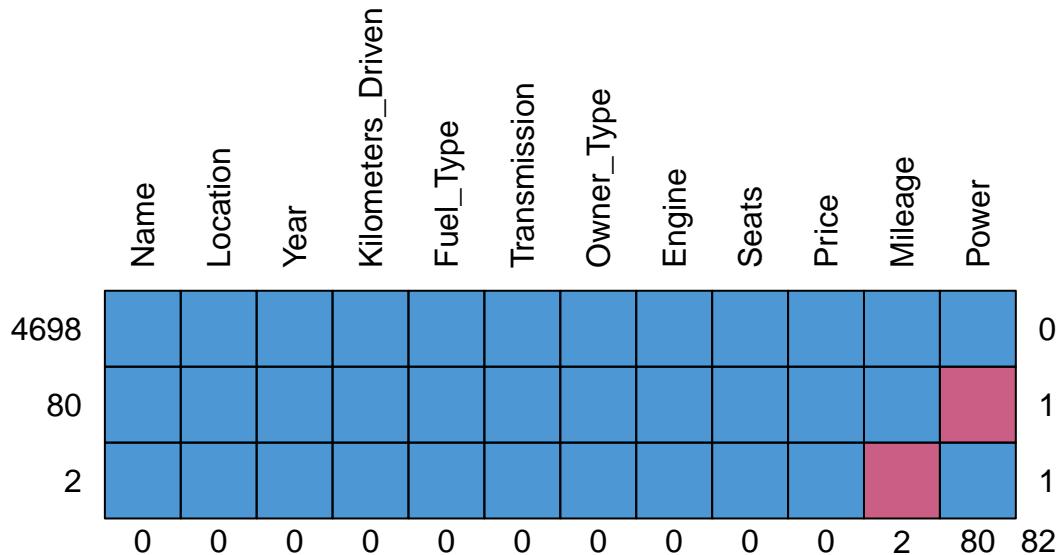
Eliminación Columnas

Eliminamos las columnas New_Price y X

```
train %>% select(-New_Price) -> train  
train %>% select(-X) -> train
```

Podemos volver a analizar los valores faltantes ahora que hemos eliminado la columna New_Price

```
md.pattern(train, plot = TRUE, rotate.names=TRUE)
```

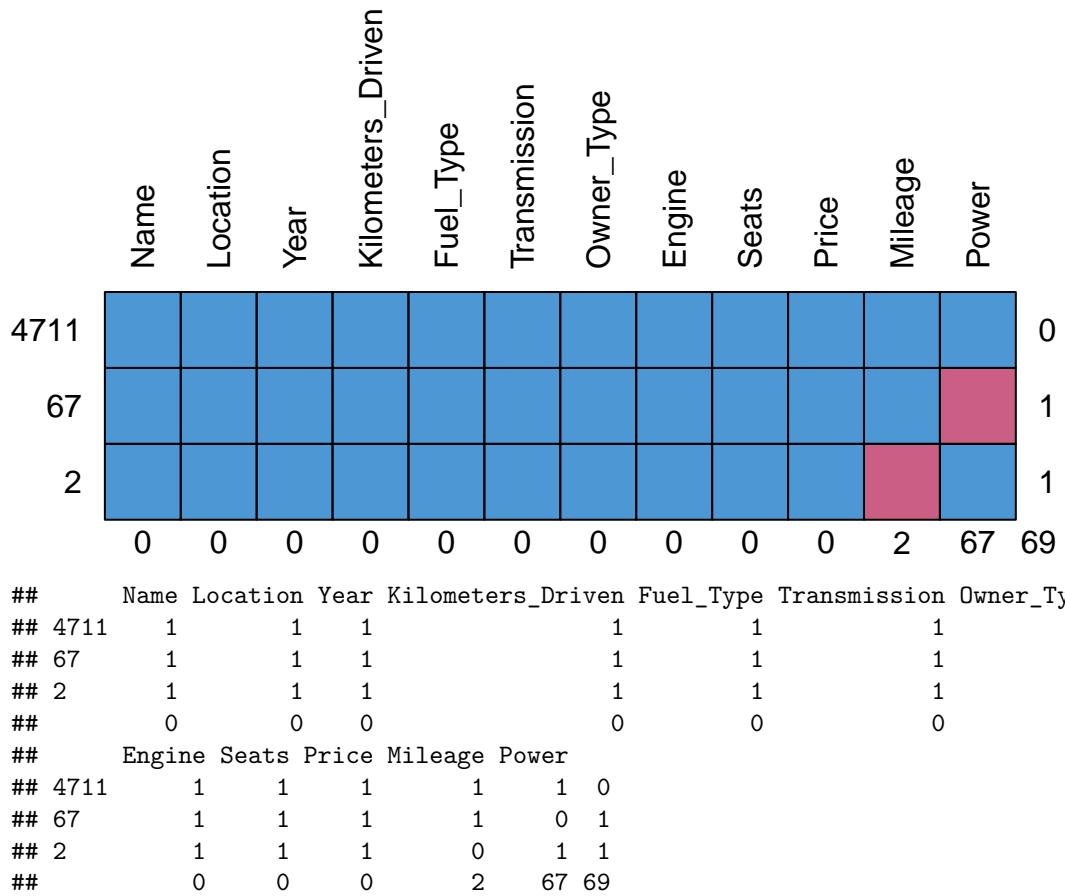


Como podemos ver ahora: -85 coches que tienen valores faltantes en las columnas Power -2 coches que no tienen datos de Mileage

Datos faltantes

Los datos faltantes se podrían llenar buscando información en internet o completandolos con los datos de coches con el mismo nombre. Vamos a probar esto.

```
train %>% filter(is.na(Engine) | is.na(Seats) | is.na(Power) | is.na(Mileage)) %>%  
  pull(Name) %>% unique() -> missing_cars #guardamos los coches que tienen faltantes  
train %>% group_by(Name) %>%  
  fill(Engine, Power, Seats, Mileage) %>% ungroup() -> train #sacamos de data esos coches  
  
md.pattern(train, plot = TRUE, rotate.names=TRUE)
```



Se puede apreciar que el número de valores faltantes disminuye pero no de forma significativa. Aún tenemos más de 70 coches con valores faltantes y dado que la mayoría tiene muchas especificaciones dentro de su propio nombre, buscar los valores que faltan coche por coche es tedioso y puede que no afecte mucho al resultado final del análisis. Procedemos a eleminiar todos los registros con valores faltantes en lugar de imputarlos.

```
train %>% drop_na() -> train
```

TRANSFORMACIÓN Y CREACIÓN DE VARIABLES

Variable Make

Por lo antes comentado de los nombres, vamos a crear una nueva variable llamada *Make*, la razón de crear esta variable es la de separar la Marca del tipo de coche ya que ambos estan juntos en la variable *Name*.

```

str_split(train$name, " ", simplify=TRUE) %>% subset(select=1) %>% unique() -> car_makes
car_makes[car_makes=="Land"] <- "Land Rover" #Anomalía en los datos
car_makes_regex <- paste(car_makes, collapse="|")
str_extract(train$name, car_makes_regex) -> make
train %>% mutate(Make = str_to_title(make)) -> train

```

```
#Hacemos una división de los coches por marca en cuanto a si son de gama baja, media o alta
train %>%
  mutate(

```

```

Gama = case_when(
  train$Make=="Datsun" |train$Make=="Smart" |train$Make=="Tata" |train$Make=="Fiat" |train$Make=="Chery" |
  train$Make=="Skoda" |train$Make=="Renault" |train$Make=="Ford" |train$Make=="Honda" |train$Make=="Vauxhall" |
  train$Make=="Bentley" |train$Make=="Porsche" |train$Make=="Land Rover" |train$Make=="Jaguar" |train$Make=="Volvo" |
  )
) -> train

train$Gama <- as.factor(train$Gama)

```

Variables Kmpl y Kmpkg

Al inicio detectamos que había dos unidades de medida en la variable Mileage.

Los kmpkg son en realidad una métrica para los combustibles de gas licuado. Así que no está justificado convertirlos en unidades de kmpl. Los coches eléctricos ni siquiera tienen la métrica del kilometraje en este conjunto de datos, algo que es de esperar.

Por lo tanto, vamos a crear dos variables llamadas kmpkg y kmpl, para que la variable Mileage pueda tener diferentes coeficientes para los tipos de combustible diesel/gasolina y GNC/GLP. La lógica es que kmpkg sólo tendrá valores no nulos para los coches alimentados con GNC/GPL, por lo que los coeficientes se ajustarán/aprenderán sólo de esas observaciones y viceversa.

```

train %>%
  mutate(kmpl = ifelse(Fuel_Type=="Diesel" | Fuel_Type=="Petrol", Mileage, 0)) %>%
  mutate(kmpkg = ifelse(Fuel_Type=="CNG" | Fuel_Type=="LPG", Mileage, 0)) %>%
  select(-Mileage) -> train

```

SELECCIÓN DE VARIABLES

Variable Respuesta

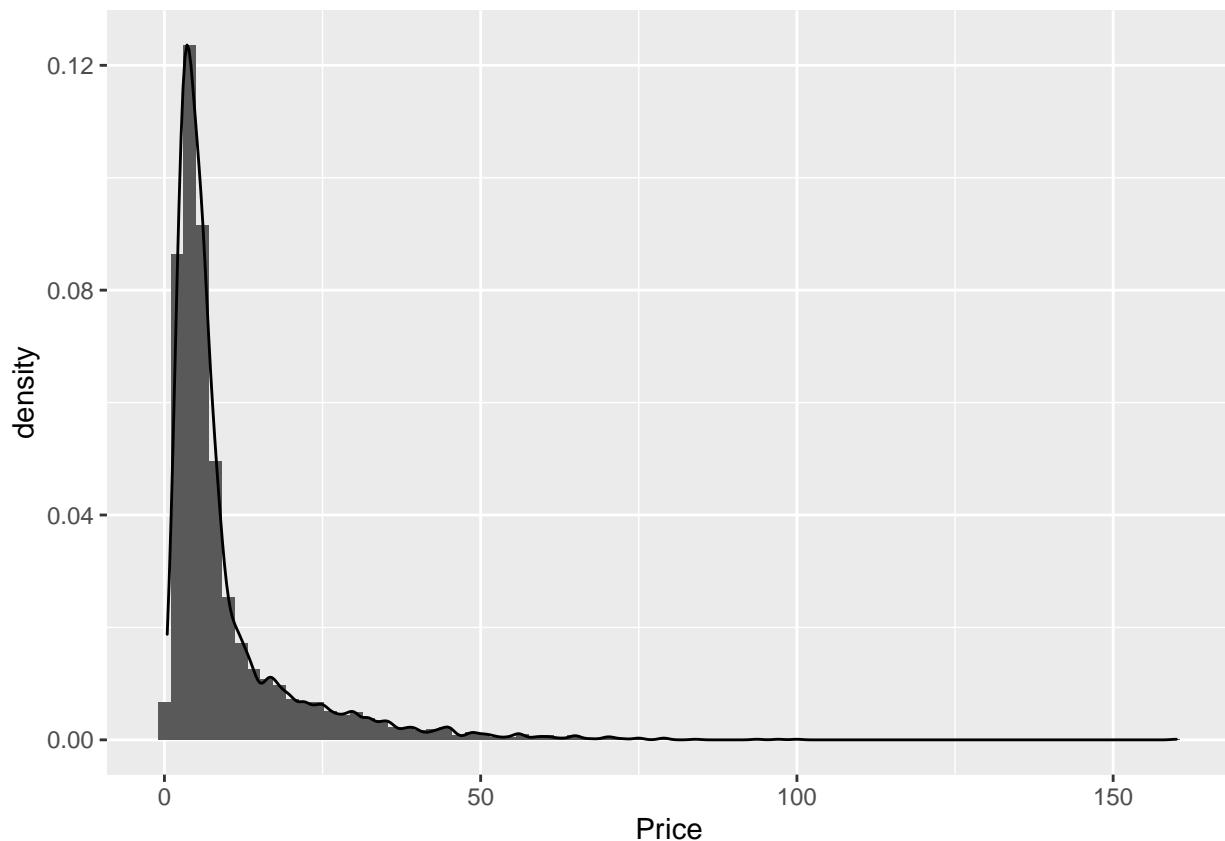
Nuestra variable respuesta va a ser: *Price*.

Por lo que en primer lugar, vamos a ver la distribución de esta variable.

```

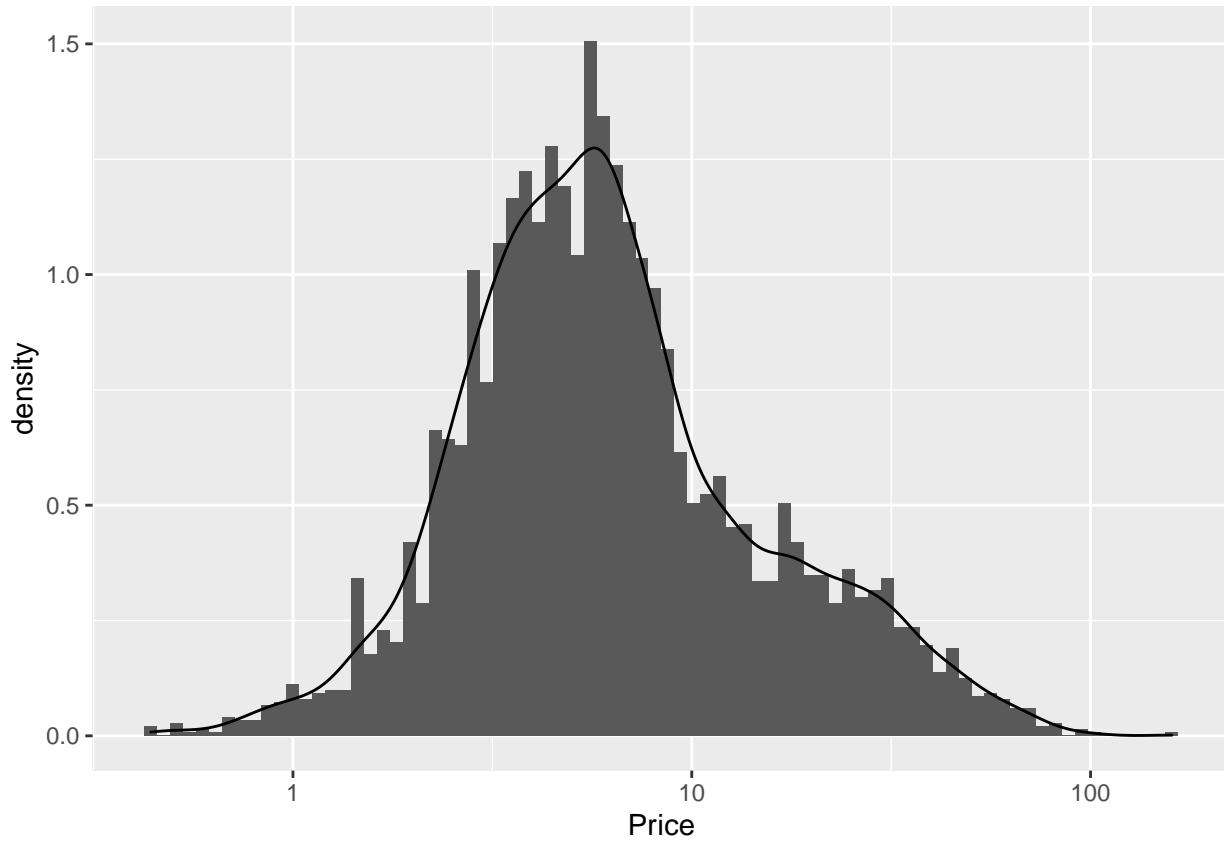
train %>% ggplot() +
  geom_histogram(aes(Price, ..density..), bins=80) +
  geom_density(aes(Price))

```



La distribución es demasiado sesgada, volvemos a probar usando una distribución logarítmica

```
train %>% ggplot() +  
  geom_histogram(aes(Price, ..density..), bins=80) +  
  geom_density(aes(Price)) +  
  scale_x_log10()
```



Esto remedia la asimetría, por lo que transformaremos la variable antes de modelarla. Para la exploración seguiremos utilizando la escala logarítmica, pero no transformaremos realmente la variable *Price* en el conjunto de datos.

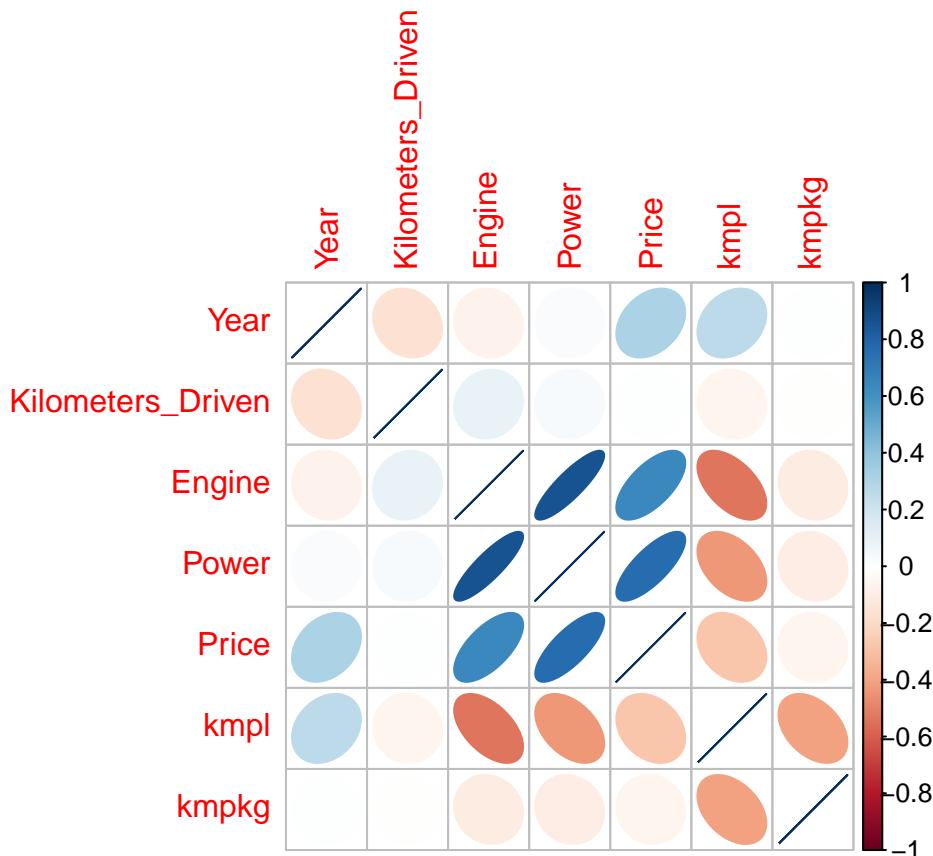
Correlación

Para analizar la correlación entre variables mostramos tres gráficos (utilizando las librerías `ellipse` y `corrplot`):

- En el primero lo que haremos será ver la correlación con elipses. Cuanto más circular sea menos correlación habrá entre las variables y cuanto más recta más correlación habrá.
- En el segundo lo que haremos será ver la correlación con una escala numérica de 0 a 1. El 0 indica ausencia de relación lineal y el 1 indica relación lineal perfecta.
- En el tercero, usando la escala que tenemos en el lateral, vemos que cuanto más oscuro y grande es el círculo, mayor relación lineal hay (azul si es positiva y rojo si es negativa), y cuanto más claro y pequeño es el círculo, menor relación lineal hay (azul si es positiva y rojo si es negativa).

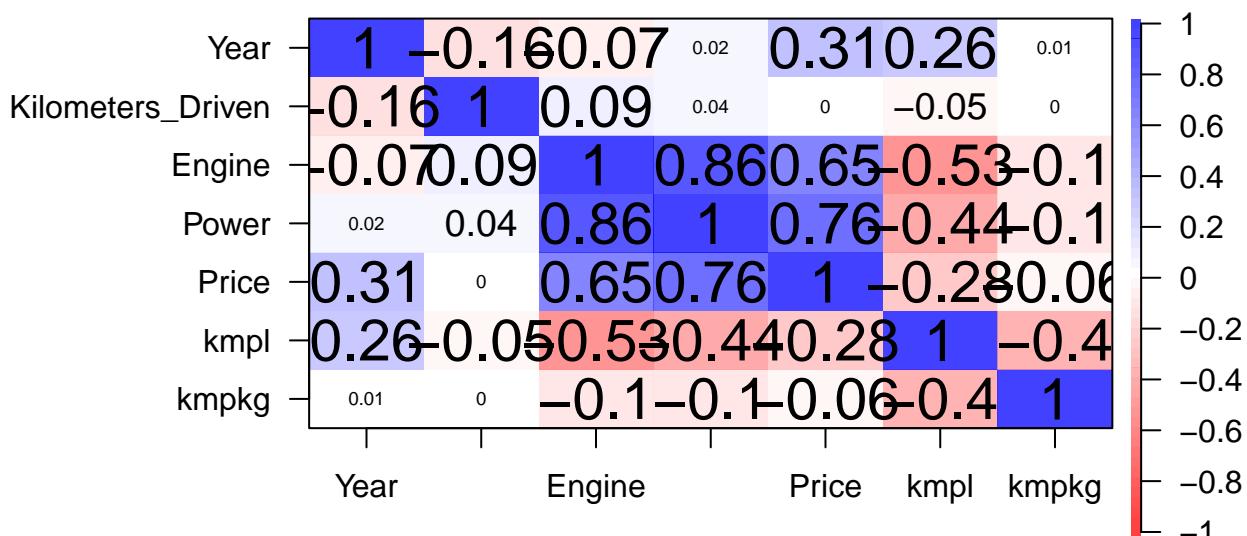
Lo hacemos únicamente, de momento, con las variables numéricas.

```
corrplot(cor(train[, c(3,4, 8, 9, 11, 14, 15)]), method = "ellipse")
```



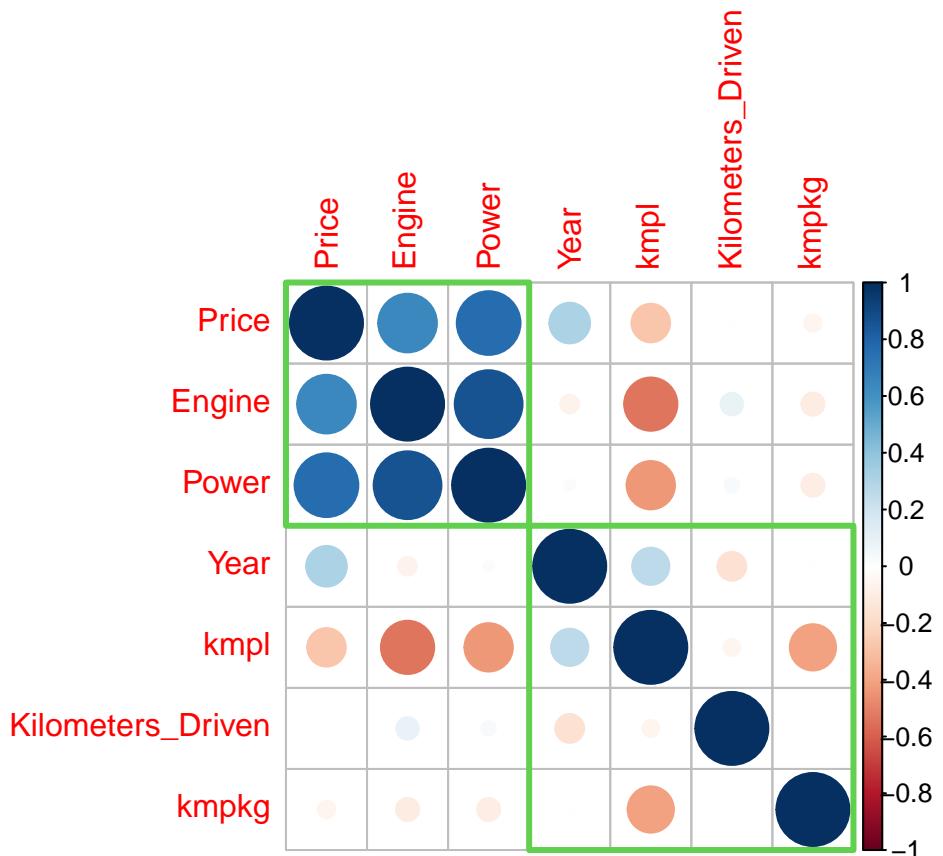
```
corPlot(train[, c(3,4, 8, 9, 11, 14, 15)], cex = 1.2, main = "Matriz de correlación")
```

Matriz de correlación



```
corrplot(cor(train[, c(3,4, 8, 9, 11, 14, 15)]), method = "circle", order = "hclust",
addrect = 2, rect.col = 3, rect.lwd = 3)
```

hclust

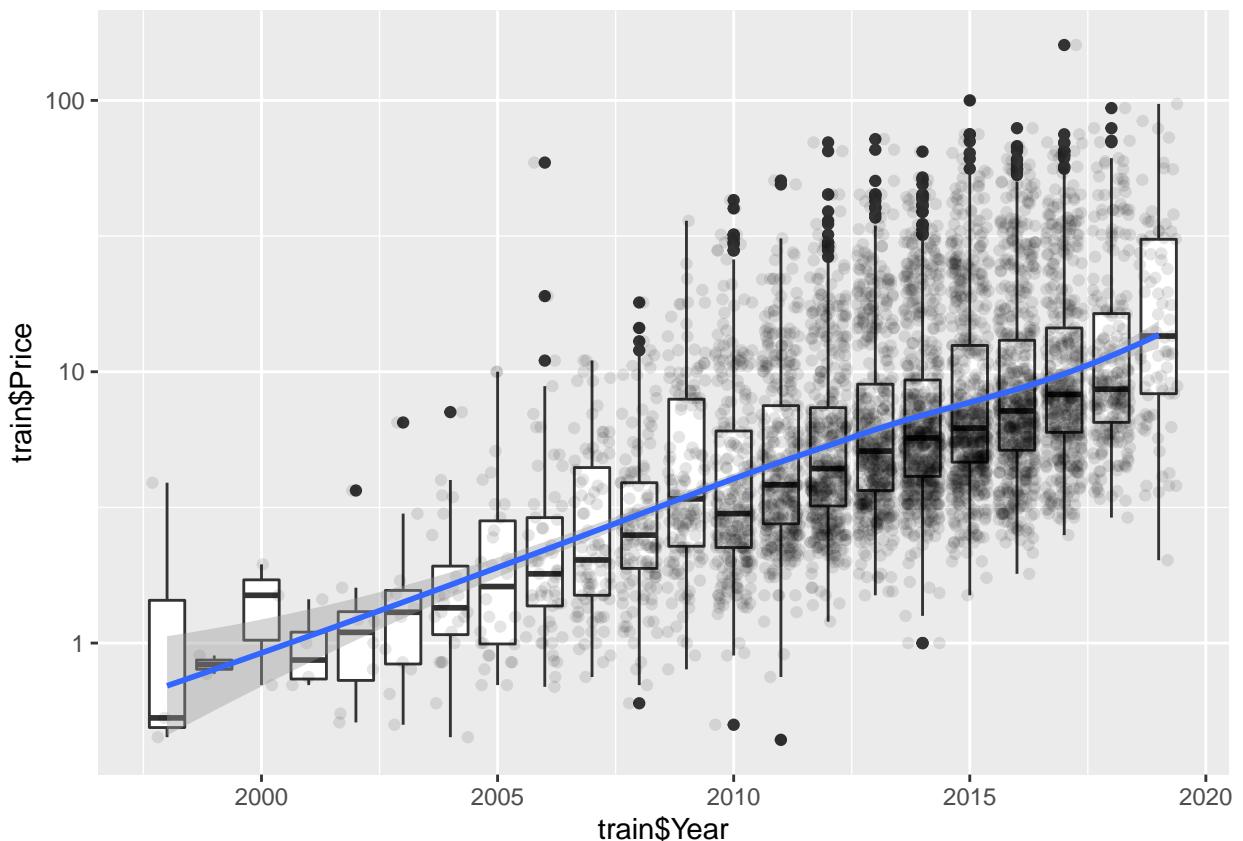


Con estos gráficos podemos ver rápidamente cuáles son las variables más correlacionadas con la variable respuesta y entre ellas mismas. Las que más pueden afectar a nuestra variable respuesta son Engine y Power. Year y kmpl tienen una correlación de 0.3 y sin embargo kmpkg no tiene nada de correlación al igual que los kilómetros recorridos que tenga el coche por lo que no servirán de mucho para nuestro análisis.

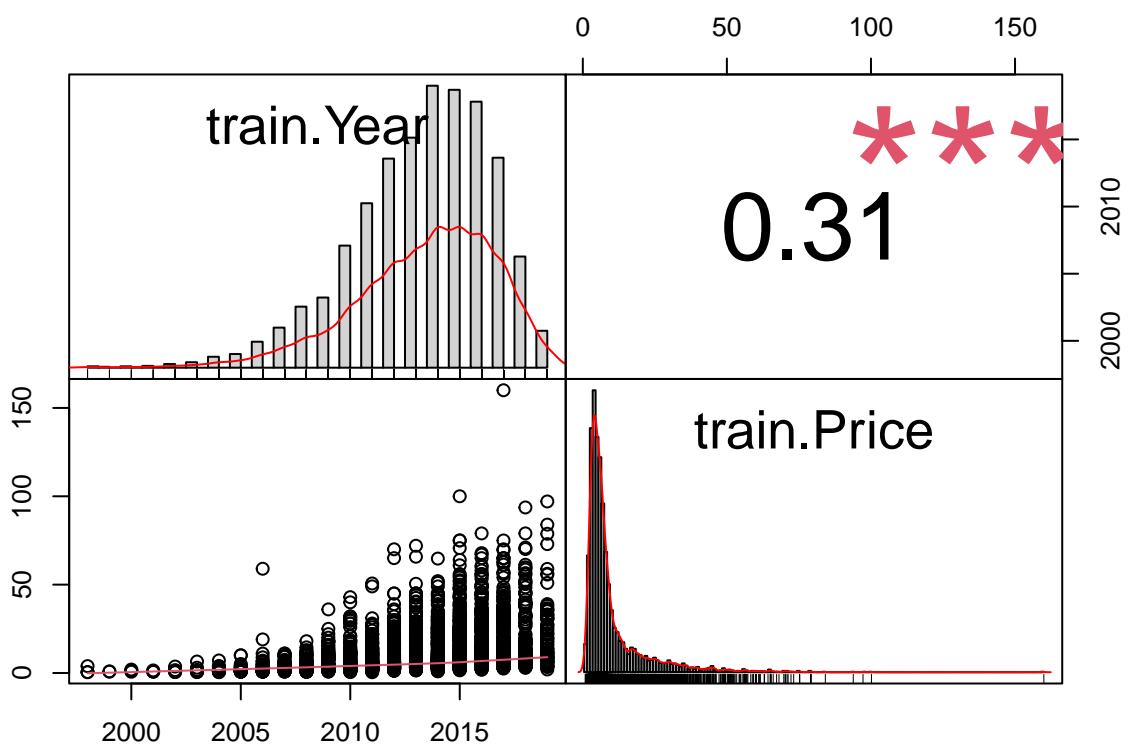
Vemos más detalladamente la correlación con cada variable, esta parte es fundamental para la selección de variables que posteriormente haremos en la regresión del modelo que creemos.

Year + Price

```
ggplot(train, aes(train$Year, train$Price)) +
  geom_boxplot(aes(group=Year)) +
  geom_jitter(alpha=0.1) +
  geom_smooth(method="loess") +
  scale_y_log10()
```



```
data.frame(train$Year, train$Price) %>% chart.Correlation()
```

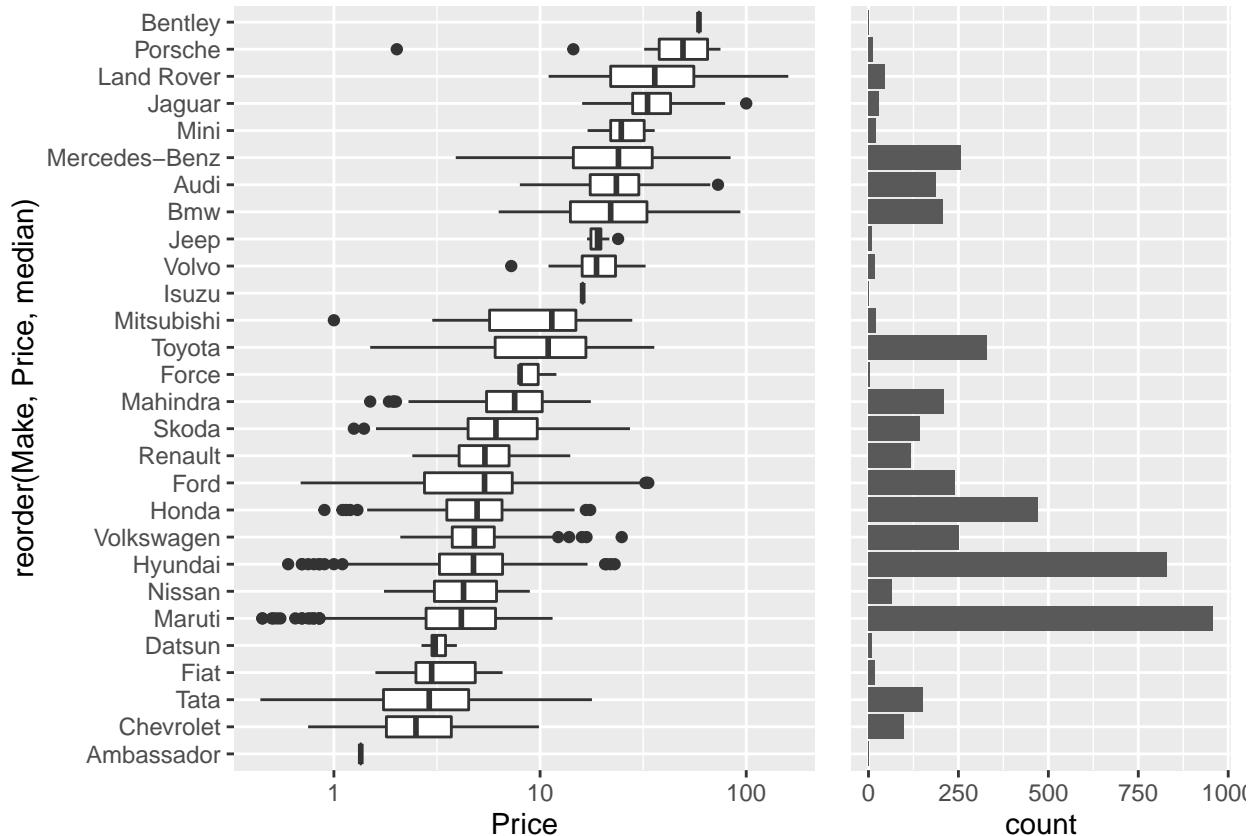


Se aprecia cierta correlación entre el año de fabricación del modelo y el precio del coche. Cuanto más nuevo es, más valor tiene. No influye de manera significativa y drástica año a año pero si aumenta, aunque sea en

poca proporción, el valor del coche.

Make + Price

```
grid.arrange(
  ggplot(train, aes(reorder(Make, Price, median), Price)) +
    geom_boxplot() +
    scale_y_log10() +
    coord_flip(),
  ggplot(train, aes(reorder(Make, Price, median))) +
    geom_bar(stat="count") +
    theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
    coord_flip(),
  ncol=3, nrow=1,
  layout_matrix=t(c(1,1,2))
)
```



Lo que podemos ver es que hay mas marcas baratas que caras. Las marcas más caras son las de coches de lujo y hay muy pocos coches de este tipo en este dataset. Por otro lado, hay muchos coches de gama media de precio decente, como Hyundai y Maruti. Esto nos da una pista de que la marca de un coche puede determinar su precio.

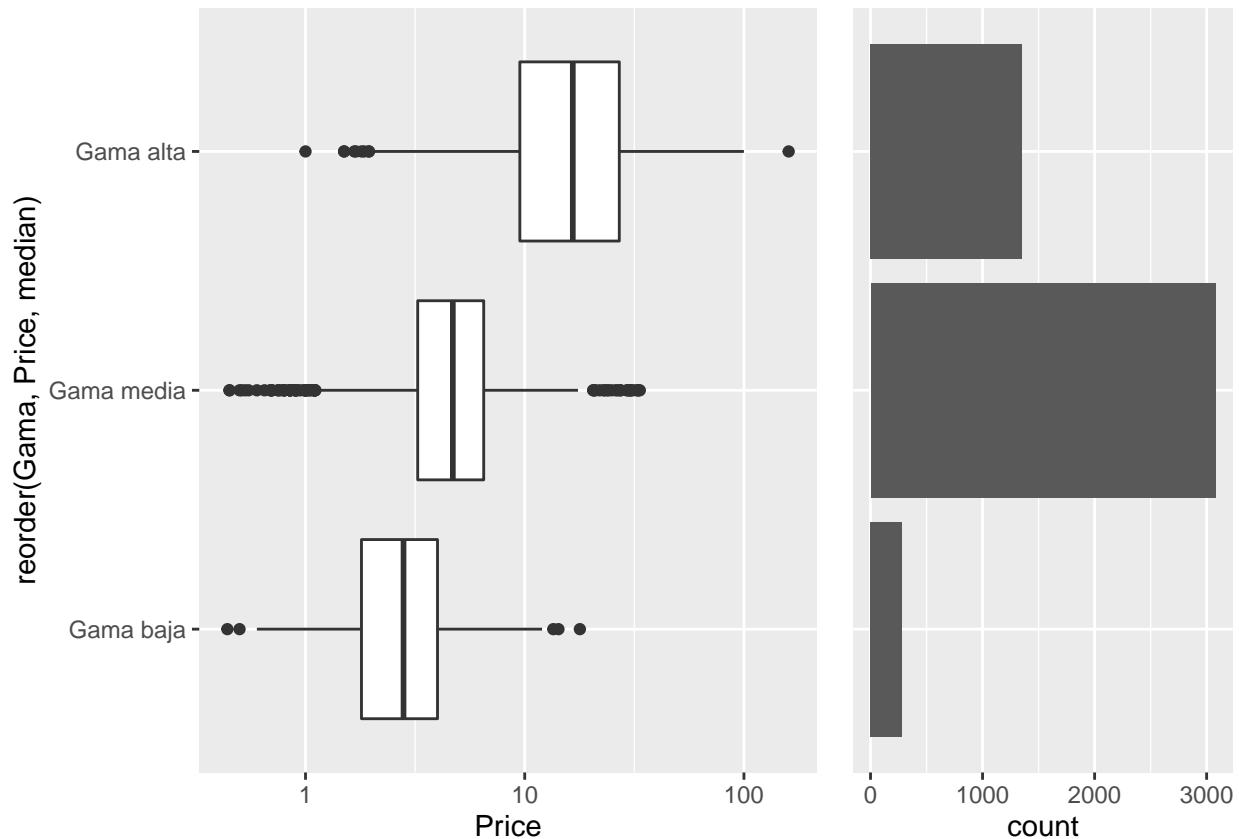
Como hemos observado que muchas de las marcas de coche seguían una distribución parecida en cuanto al precio, se han agrupado en 3 gamas: gama baja, gama media y gama alta. Así, su visualización y la posterior utilidad de la marca en el modelo de regresión lineal será más frutífera y simplificada.

```
grid.arrange(
  ggplot(train, aes(reorder(Gama, Price, median), Price)) +
    geom_boxplot() +
```

```

    scale_y_log10() +
    coord_flip(),
ggplot(train, aes(reordered(Gama, Price, median))) +
  geom_bar(stat="count") +
  theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
  coord_flip(),
  ncol=3, nrow=1,
  layout_matrix=t(c(1,1,2))
)

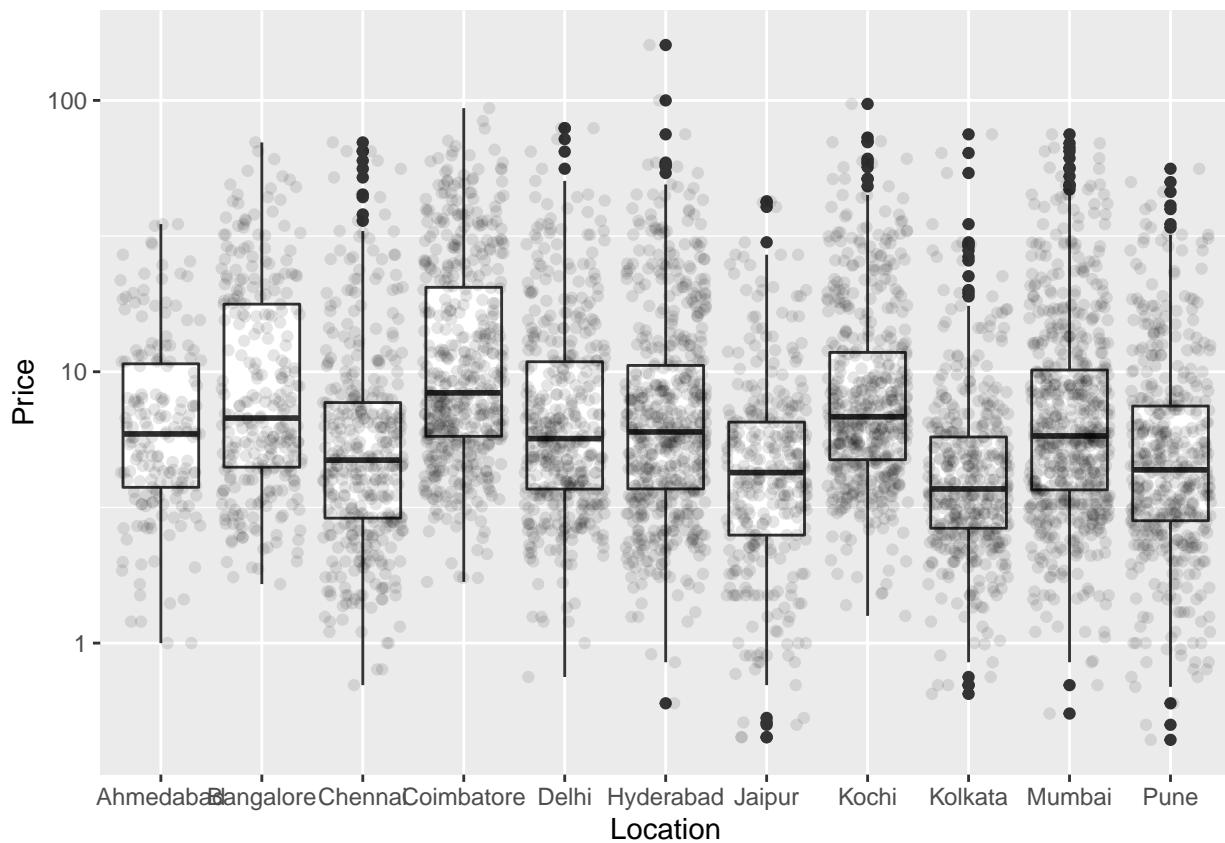
```



Los coches de gama media representan la mayoría de muestras en nuestro dataset. Después, los de gama alta y por último los que menos relevancia tienen en nuestros datos, de momento, son los de gama baja. Esta agrupación favorece tanto la representación de los datos de manera más ordenada como la visualización de los mismos. Además, no tendría sentido meter tantas marcas en un modelo de regresión lineal de manera unitaria.

Location + Price

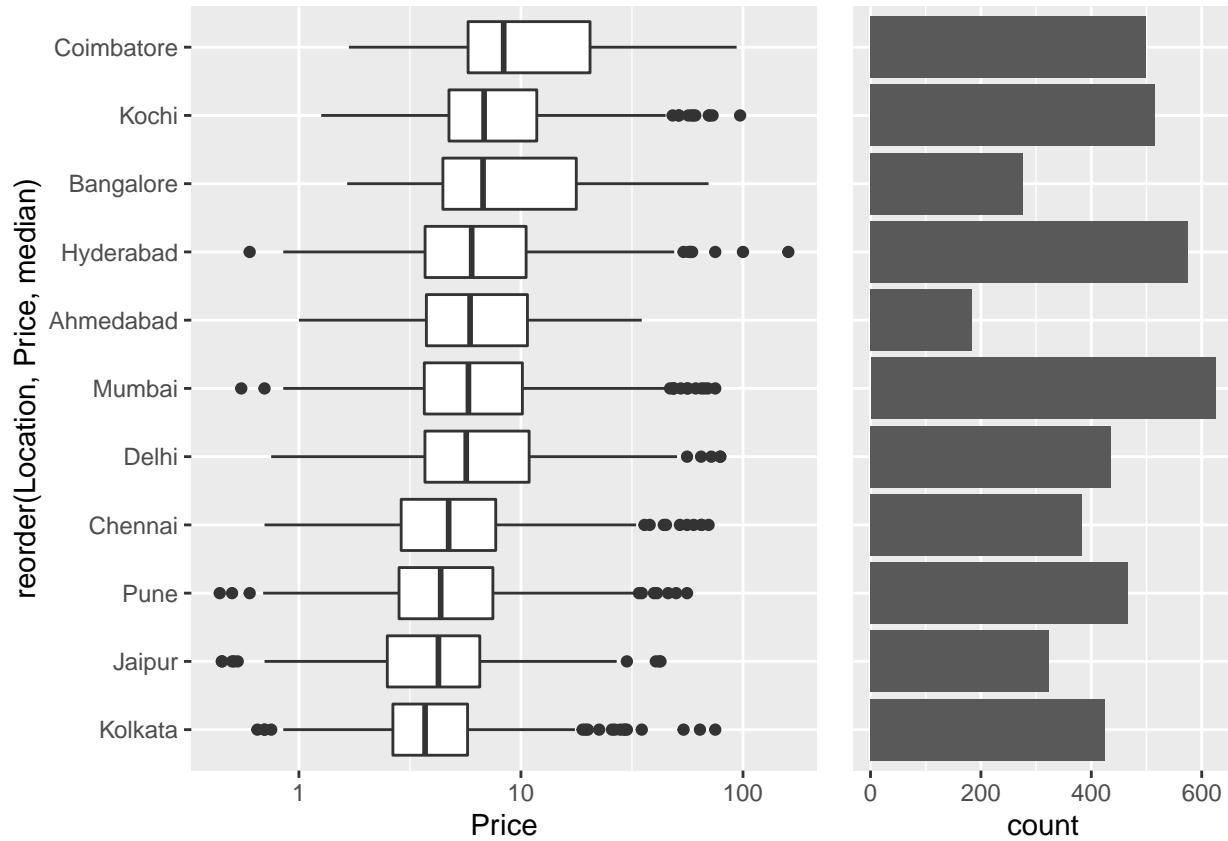
```
ggplot(train, aes(Location, Price)) + geom_boxplot(aes(group=Location)) + geom_jitter(alpha=0.1) + geom_smooth()
```



```

grid.arrange(
  ggplot(train, aes(reorder(Location, Price, median), Price)) +
    geom_boxplot() +
    scale_y_log10() +
    coord_flip(),
  ggplot(train, aes(reorder(Location, Price, median))) +
    geom_bar(stat="count") +
    theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
    coord_flip(),
    ncol=3, nrow=1,
    layout_matrix=t(c(1,1,2))
)

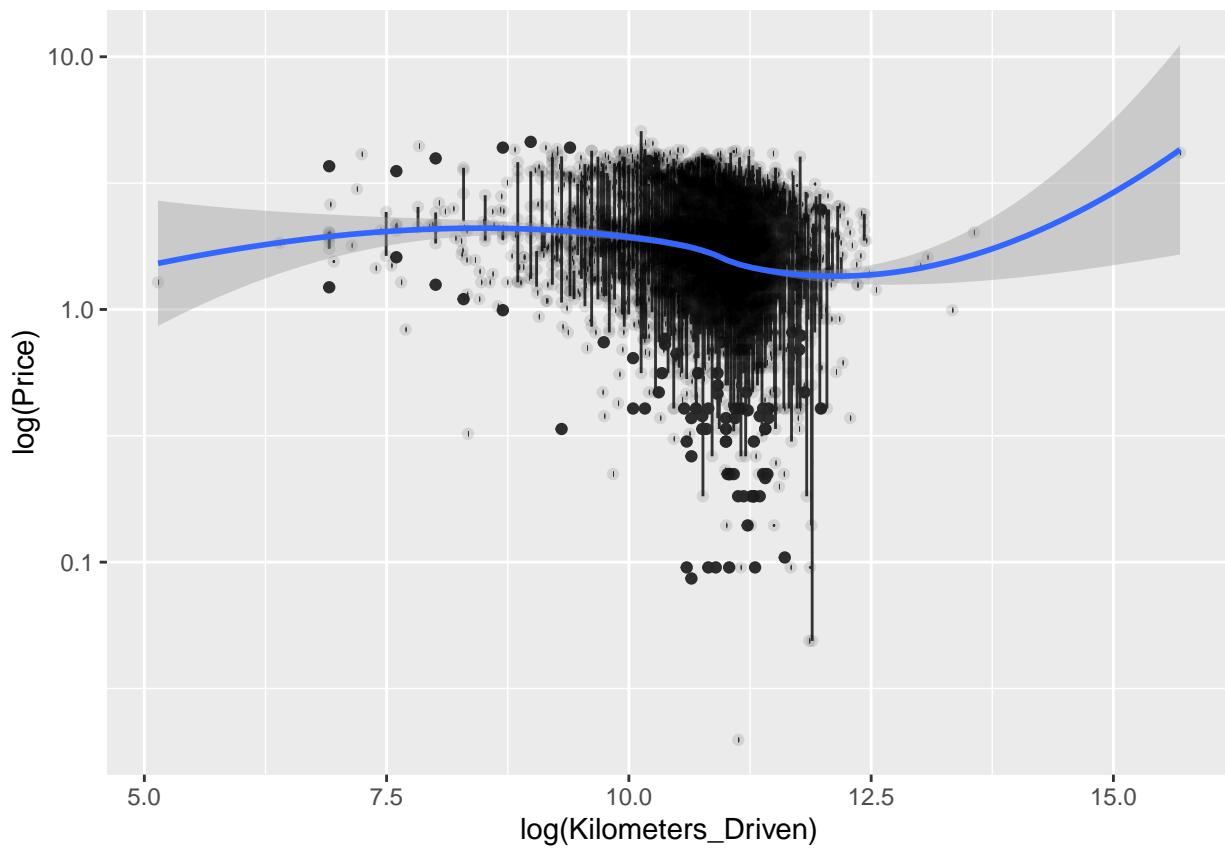
```



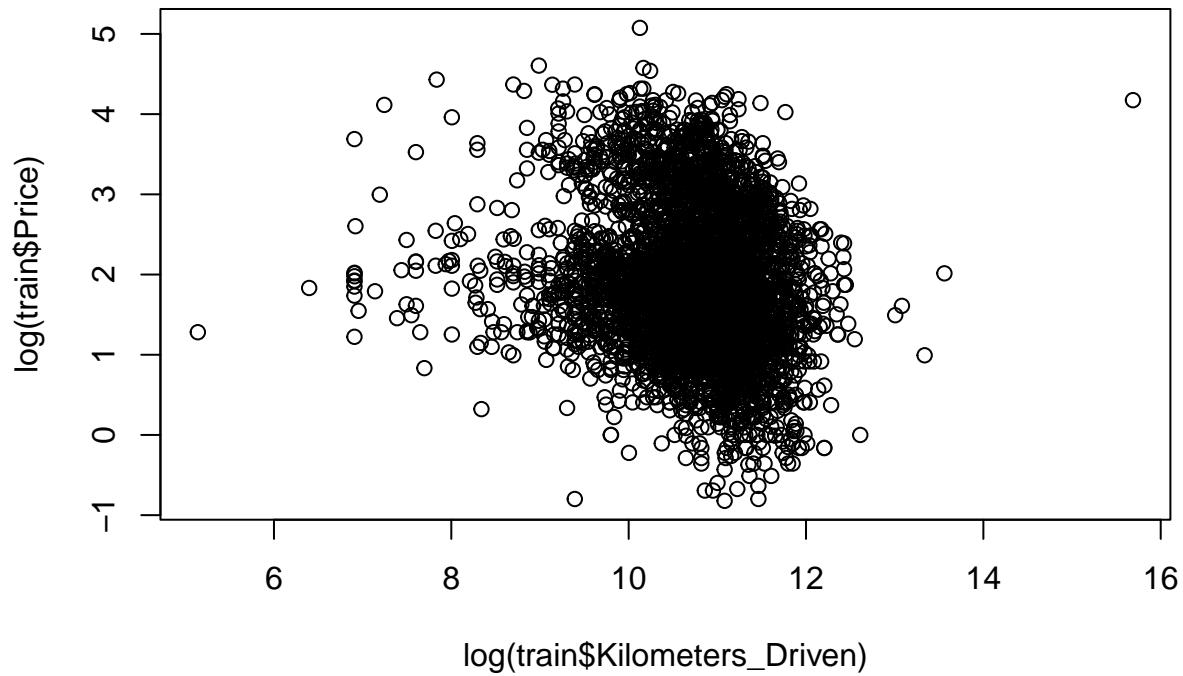
Los coches disponibles están bien distribuidos en cada una de las ciudades de la India y como se aprecia, no tiene mucha relación el lugar de venta con el precio que cuesta.

Kilometers Driven + Price

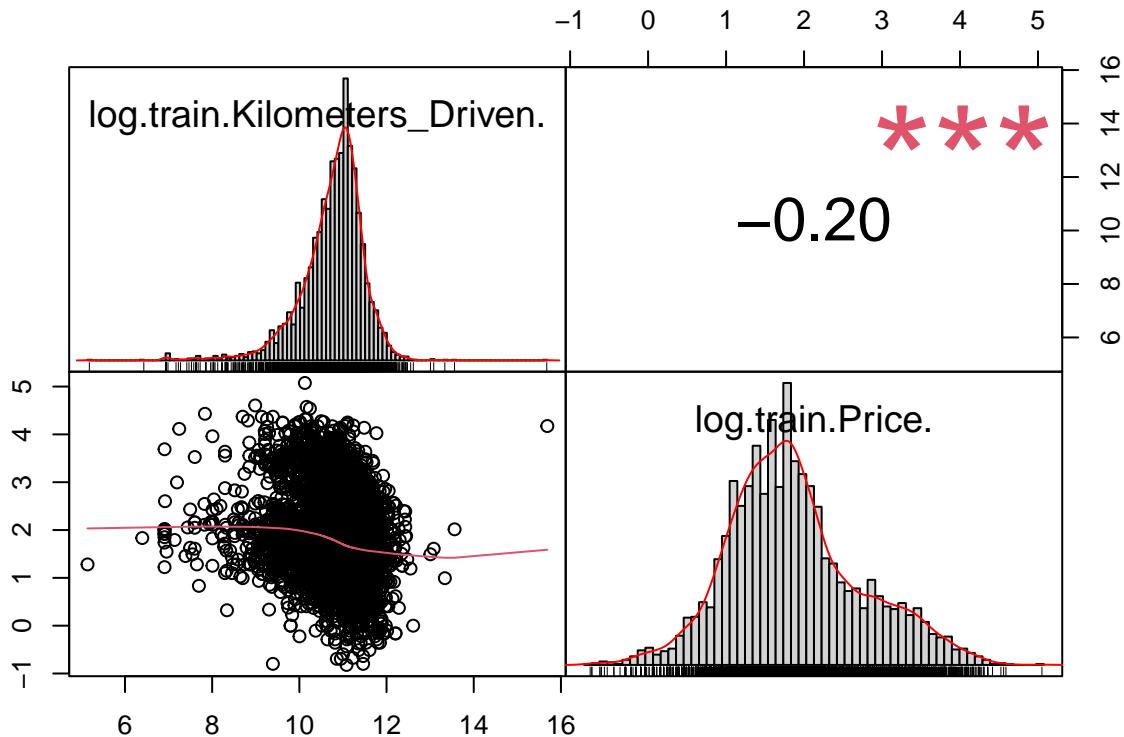
```
ggplot(train, aes(log(Kilometers_Driven), log(Price)))+
  geom_boxplot(aes(group=Kilometers_Driven))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```



```
plot(log(train$Kilometers_Driven), log(train$Price))
```



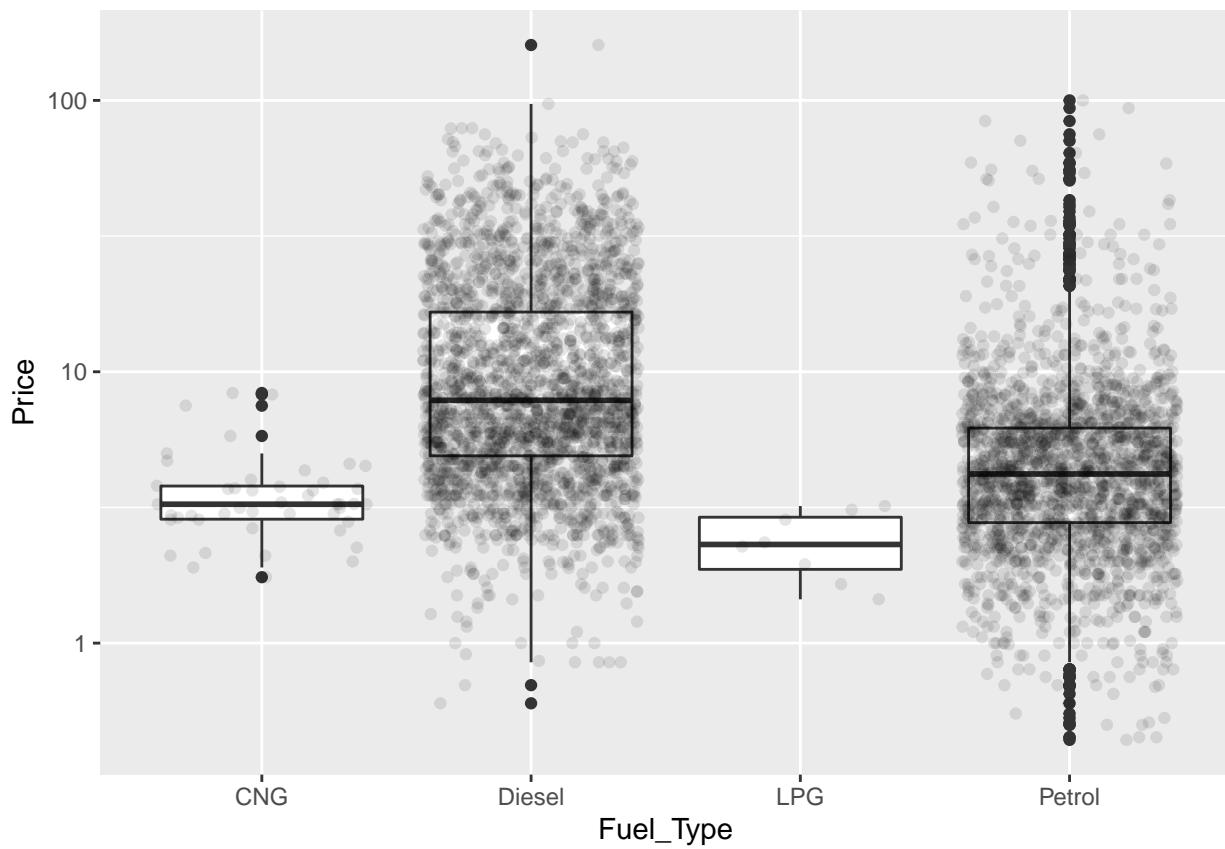
```
data.frame(log(train$Kilometers_Driven), log(train$Price)) %>%
  chart.Correlation()
```



La empresa de ventas de coches parece haber decidido que no es relevante el número de kilómetros recorridos con respecto al precio. Mientras el coche circule y esté en óptimas condiciones, no se tiene en cuenta el número de kilómetros que suele estar relacionado con el año de fabricación del coche.

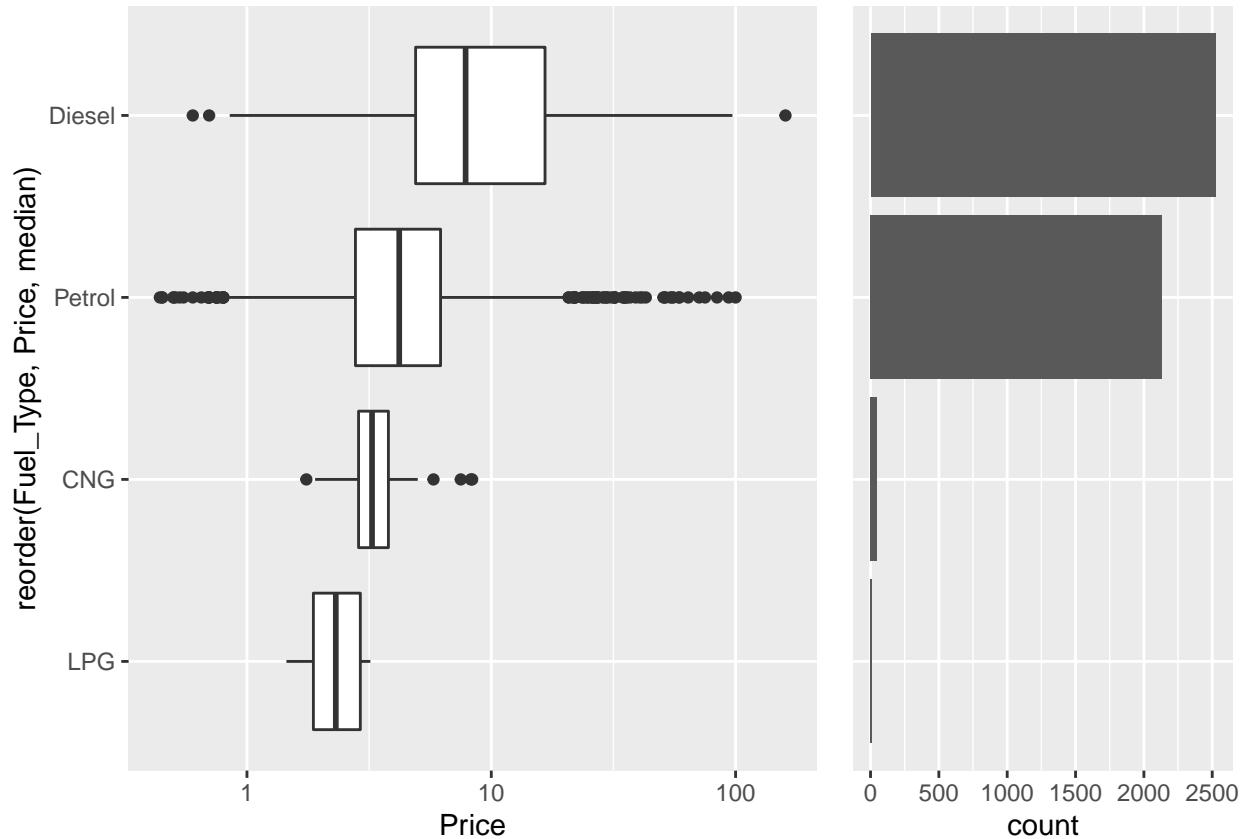
Fuel Type + Price

```
ggplot(train, aes(Fuel_Type, Price))+
  geom_boxplot(aes(group=Fuel_Type))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```



```

grid.arrange(
  ggplot(train, aes(reorder(Fuel_Type, Price, median), Price)) +
    geom_boxplot() +
    scale_y_log10() +
    coord_flip(),
  ggplot(train, aes(reorder(Fuel_Type, Price, median))) +
    geom_bar(stat="count") +
    theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
    coord_flip(),
    ncol=3, nrow=1,
    layout_matrix=t(c(1,1,2))
)
  
```



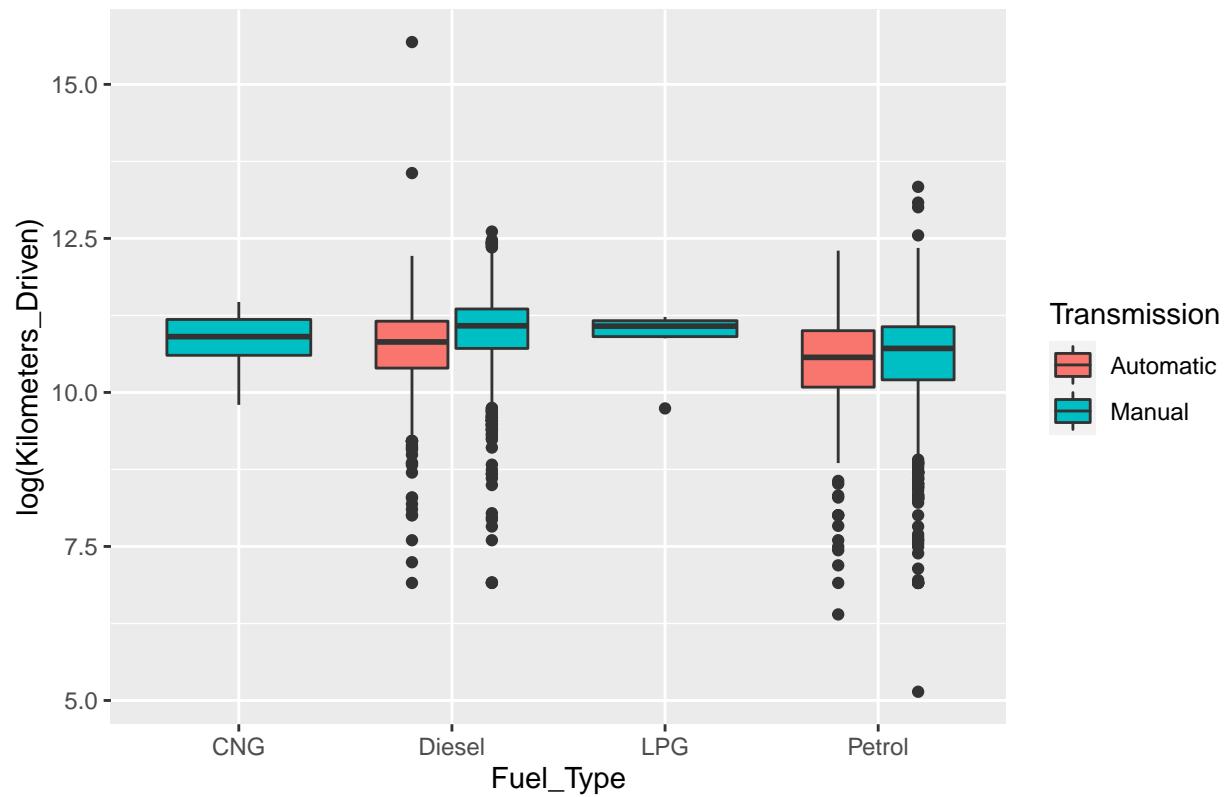
Lo primero que se aprecia es que la mayoría de coches son de diésel y gasolina y además son los que mayor precio tienen. Un coche diésel es más caro que uno de gasolina generalmente básicamente por el consumo que realiza cada uno de ellos.

Pero, tiene relación el tipo de combustible con los kilómetros recorridos?

Fuel_Type + Kilometers_driven

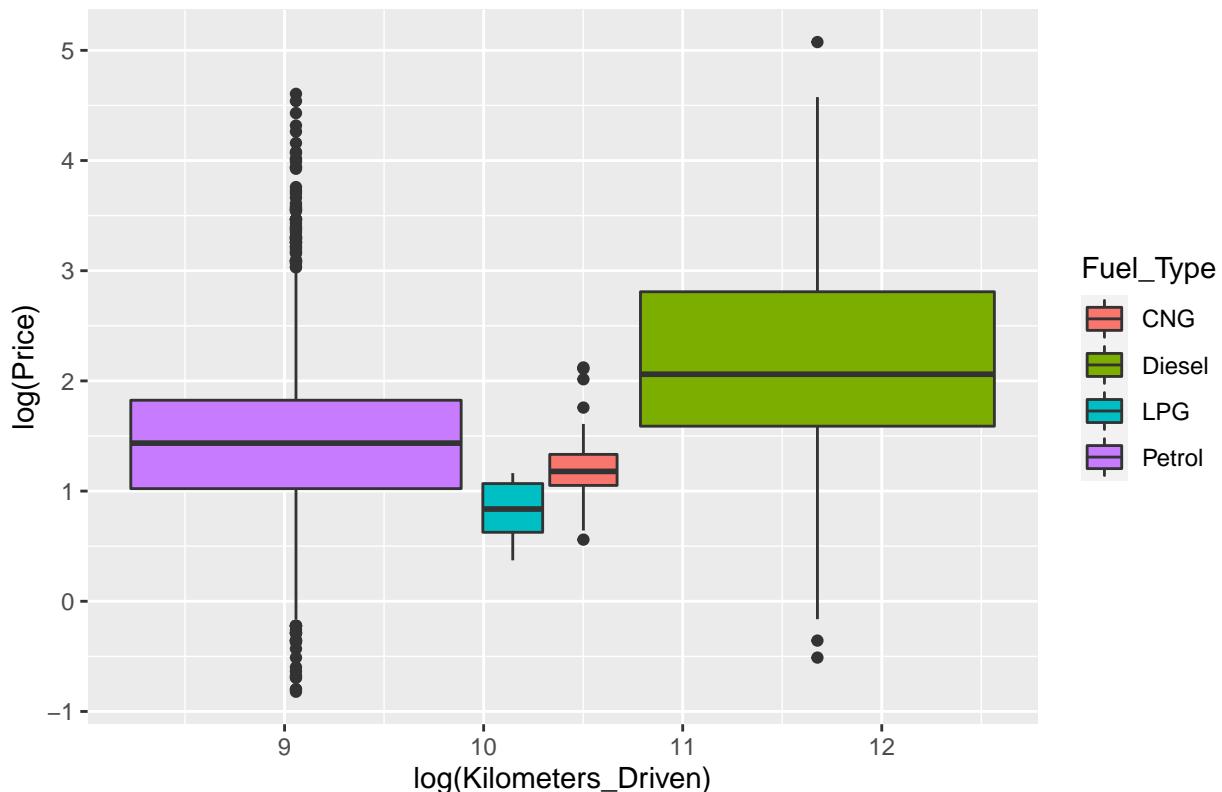
```
train %>% ggplot(aes(x=Fuel_Type, y=log(Kilometers_Driven), fill=Transmission)) +geom_boxplot() +ggtile
```

Km recorridos según el tipo de combustible por tipo de transmisión



```
train %>% ggplot(aes(x=log(Kilometers_Driven), y=log(Price), fill=Fuel_Type)) +geom_boxplot() +ggtitle (
```

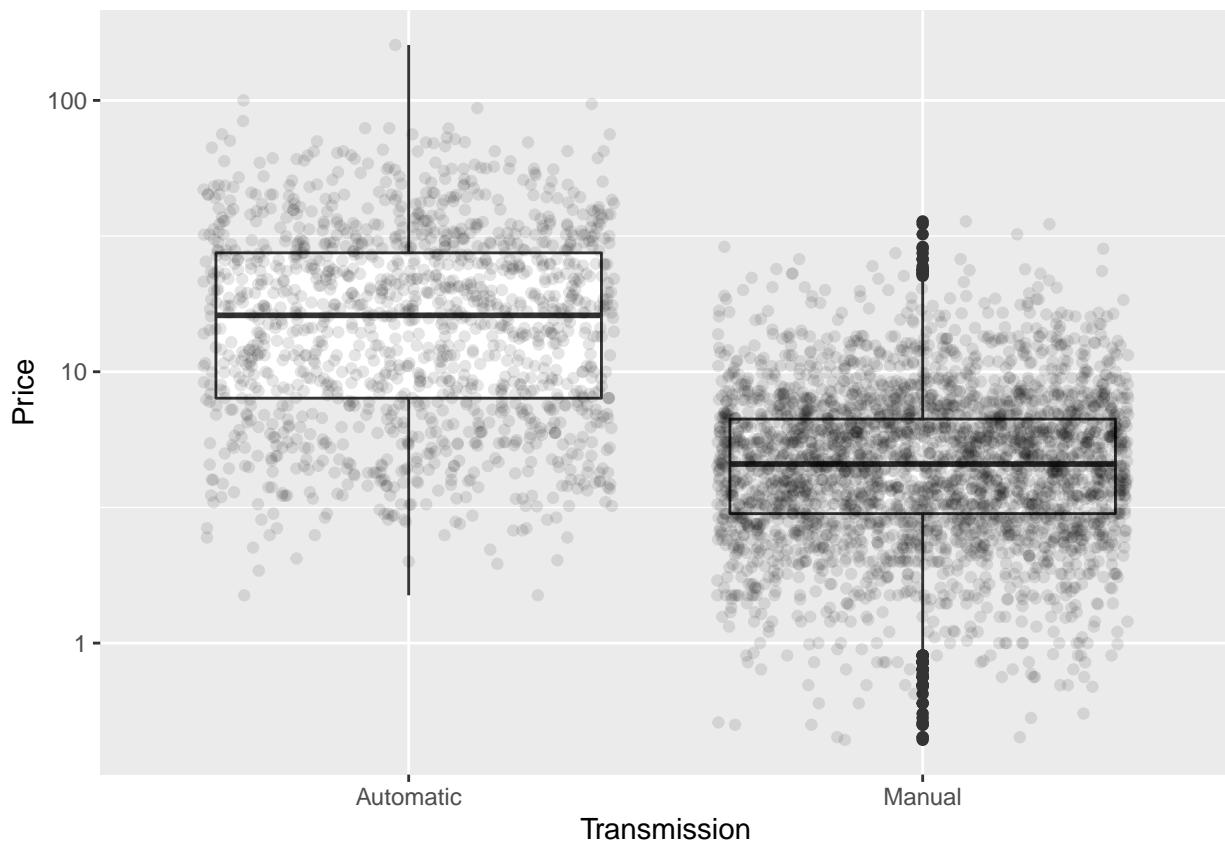
Precio según los km recorridos por tipo de combustible



Con esto podemos apreciar que los kilómetros que recorramos sí importa en qué tipo de coche lo hagamos puesto que no es lo mismo recorrer 100.000 km con un coche Diésel que con un coche de gasolina. Un factor clave que a menudo hace caer la balanza del lado de los diésel es que la mayoría de compradores prefiere gastar más en la adquisición de un coche diésel, que como podemos ver en la gráfica, son más caros, que pagar más en los repostajes por haber comprado un gasolina. Es como si uno hiciera una inversión mayor al principio y luego ya fuese toda una vida útil del coche de ahorro.

Transmission + Price

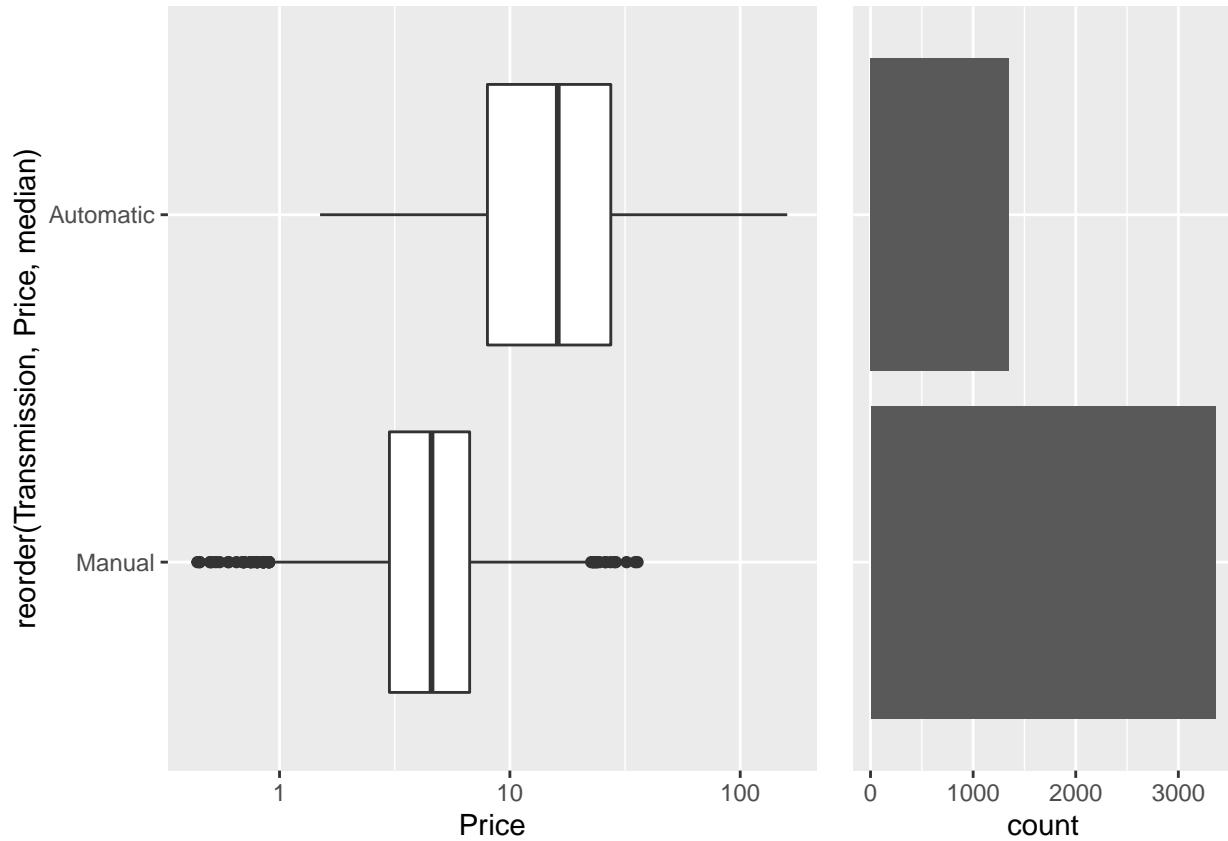
```
ggplot(train, aes(Transmission, Price))+
  geom_boxplot(aes(group=Transmission))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```



```

grid.arrange(
  ggplot(train, aes(reorder(Transmission, Price, median), Price)) +
    geom_boxplot() +
    scale_y_log10() +
    coord_flip(),
  ggplot(train, aes(reorder(Transmission, Price, median))) +
    geom_bar(stat="count") +
    theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
    coord_flip(),
    ncol=3, nrow=1,
    layout_matrix=t(c(1,1,2))
)

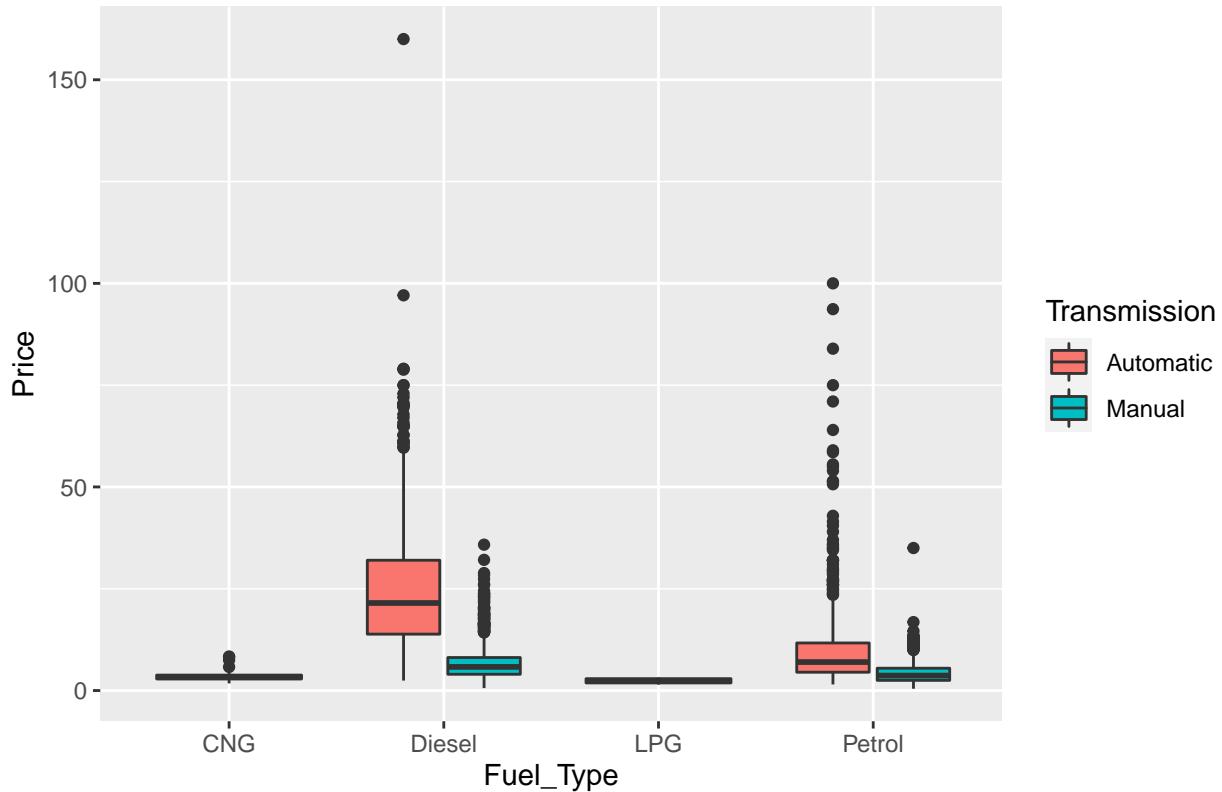
```



En este dataset hay más coches manuales que automáticos como se aprecia pero los automáticos son más caros. Esto es porque ofrecen mayor comodidad, la conducción es más segura y atenta al tráfico al no tener que estar pendiente del pedal y la palanca, aparte de la diferencia de consumo entre ambos.

```
train %>% ggplot(aes(x=Fuel_Type, y=Price, fill=Transmission)) +geom_boxplot() +ggttitle ("Precio de los coches")
```

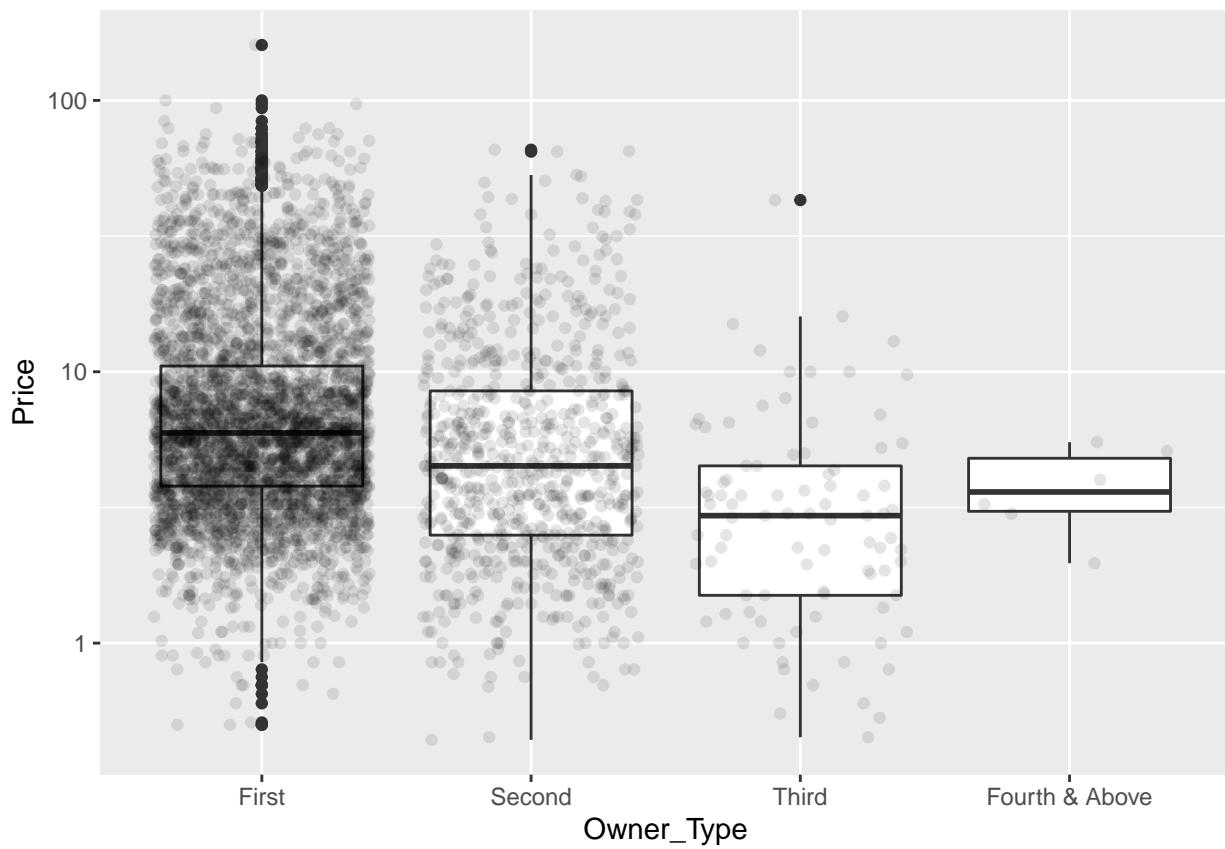
Precio de los coches según el tipo de combustible por tipo de trasmisión



Una vez más, esto confirmó que el vehículo de gasolina es más barato que el diésel.

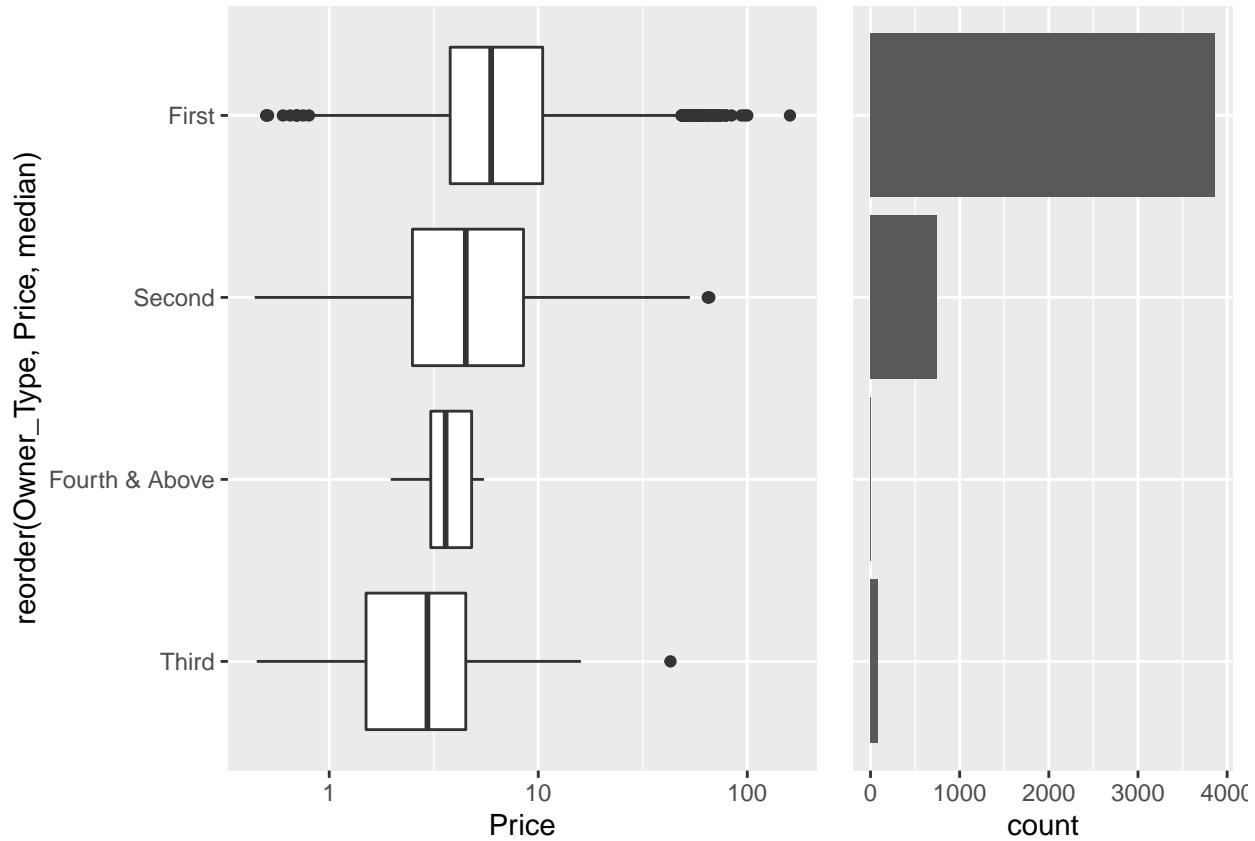
Owner Type + Price

```
ggplot(train, aes(Owner_Type, Price)) + geom_boxplot(aes(group=Owner_Type)) + geom_jitter(alpha=0.1) + geom_s
```



```

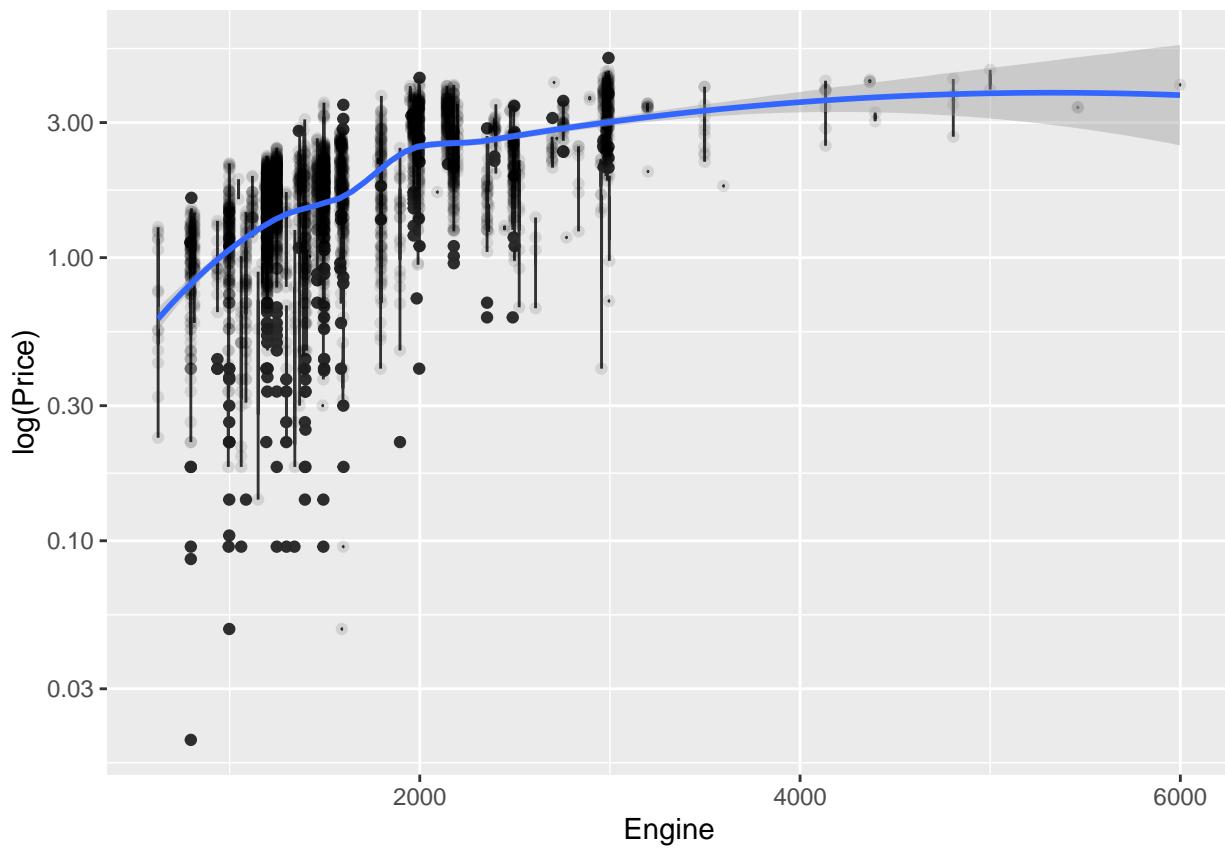
grid.arrange(
  ggplot(train, aes(reorder(Owner_Type, Price, median), Price)) +
    geom_boxplot() +
    scale_y_log10() +
    coord_flip(),
  ggplot(train, aes(reorder(Owner_Type, Price, median))) +
    geom_bar(stat="count") +
    theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
    coord_flip(),
    ncol=3, nrow=1,
    layout_matrix=t(c(1,1,2))
)
  
```



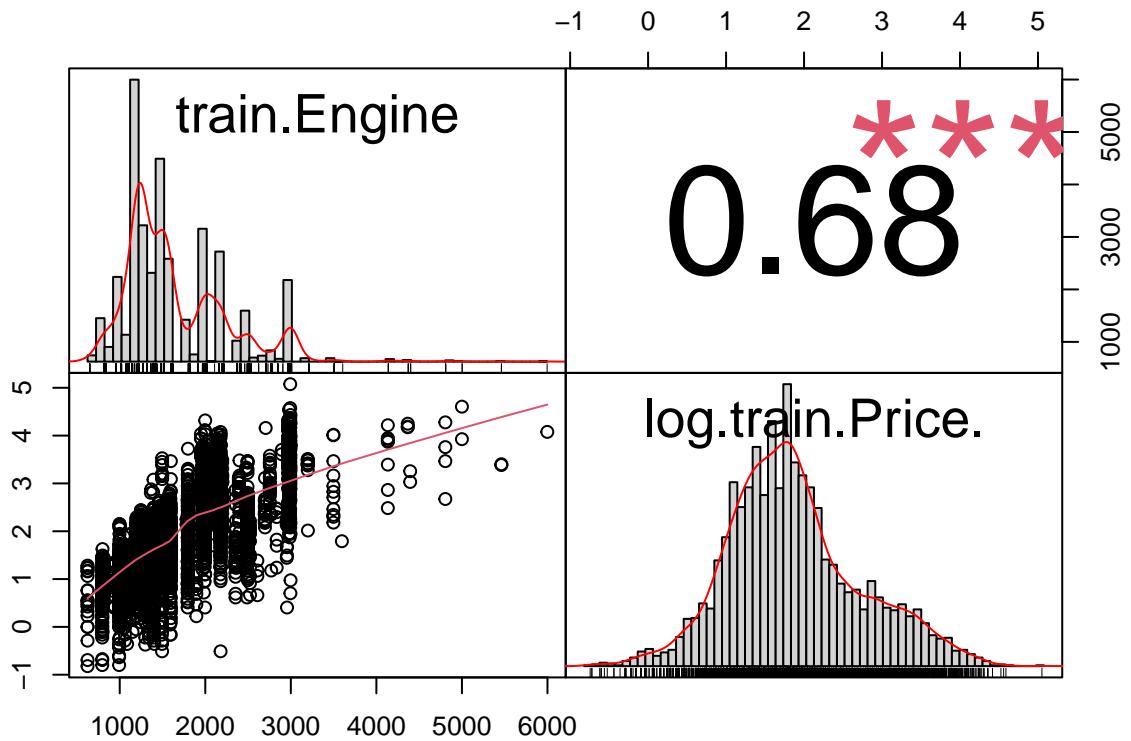
La mayoría de coches de segunda mano son los que han pasado ya por un conductor o en su defecto por dos. Antigüedad de 3 dueños es poco común y se da pocas veces generalmente. Está estrechamente relacionado el precio con los dueños que ha tenido el coche, siendo más caros los que menos dueños han tenido, lógicamente.

Engine + Price

```
ggplot(train, aes(Engine, log(Price)))+
  geom_boxplot(aes(group=Engine))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```



```
data.frame(train$Engine, log(train$Price)) %>% chart.Correlation()
```

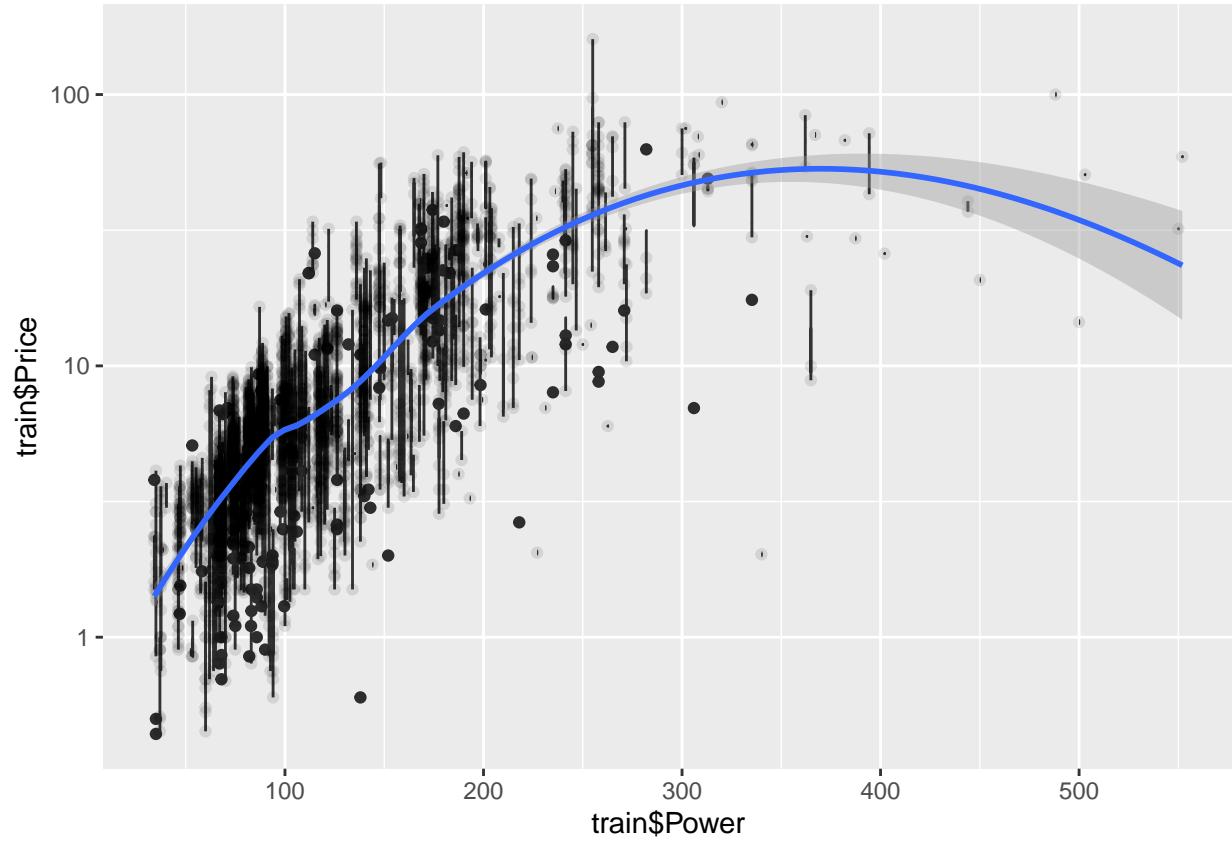


Atendiendo a los gráficos se tiene una alta correlación entre la cilindrada del motor y el precio del coche. Cuanto mayor sea la cilindrada del motor, mayor será el precio del coche ya que es un factor determinante en

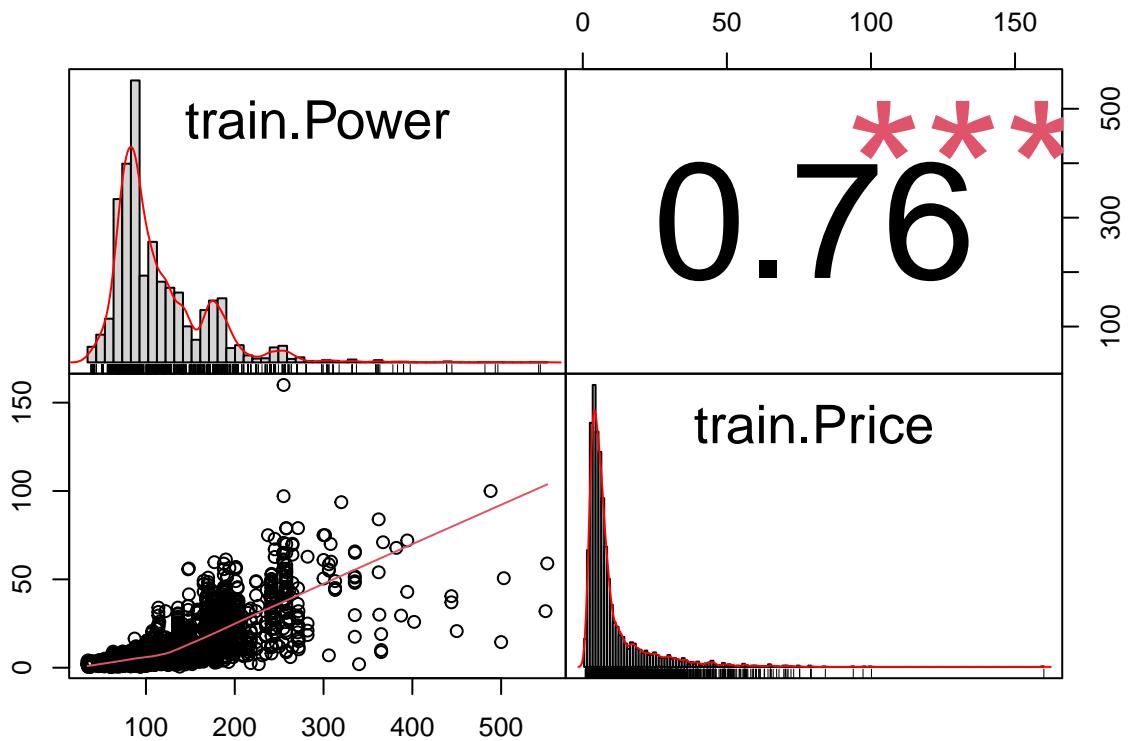
la potencia del coche y en el combustible que éste consume.

Power + Price

```
ggplot(train, aes(train$Power, train$Price))+
  geom_boxplot(aes(group=train$Power))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```



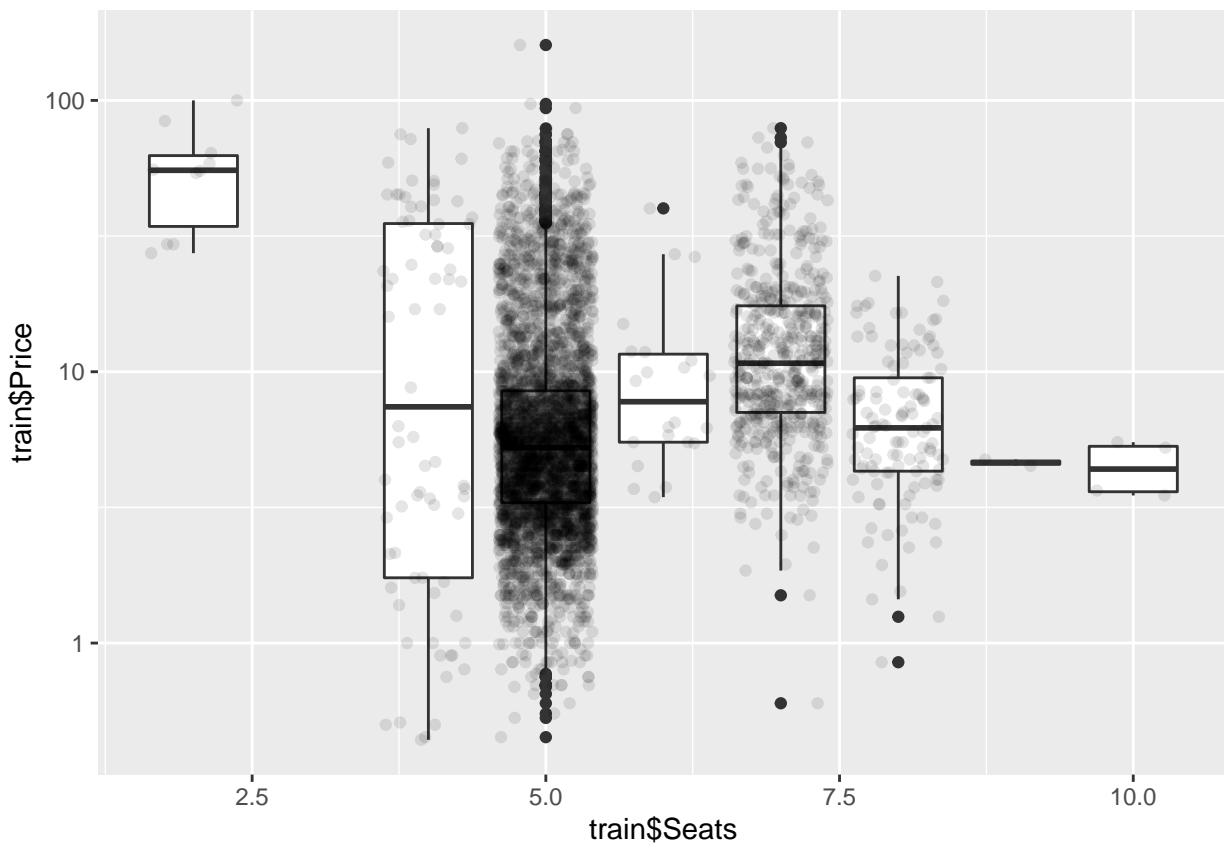
```
data.frame(train$Power, train$Price) %>% chart.Correlation()
```



La potencia del motor está relacionada altamente y de manera lineal con el precio del coche. A mayor potencia, mayor será el precio de compra.

Seats + Price

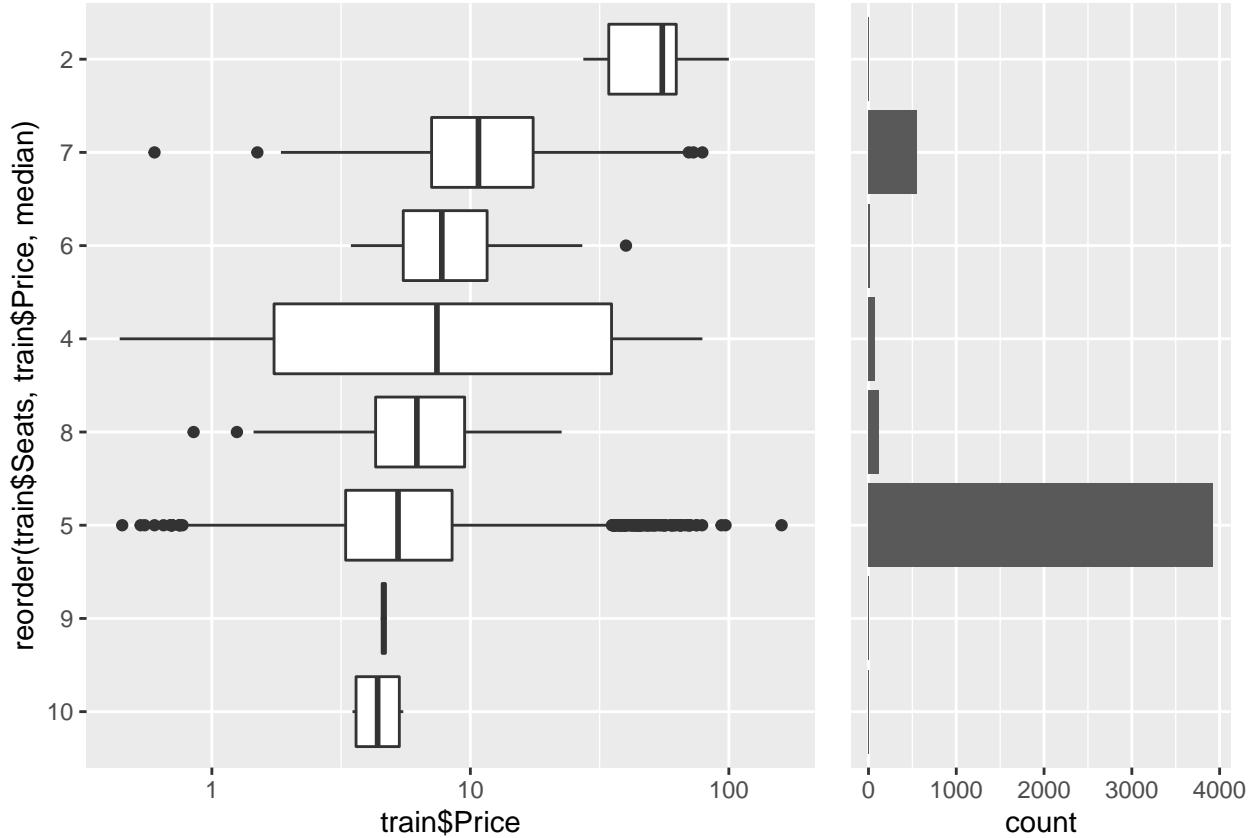
```
ggplot(train, aes(train$Seats, train$Price))+
  geom_boxplot(aes(group=train$Seats))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```



```

grid.arrange(
  ggplot(train, aes(reorder(train$Seats, train$Price, median), train$Price)) +
    geom_boxplot() +
    scale_y_log10() +
    coord_flip(),
  ggplot(train, aes(reorder(train$Seats, train$Price, median))) +
    geom_bar(stat="count") +
    theme(axis.title.y=element_blank(), axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
    coord_flip(),
  ncol=3, nrow=1,
  layout_matrix=t(c(1,1,2))
)

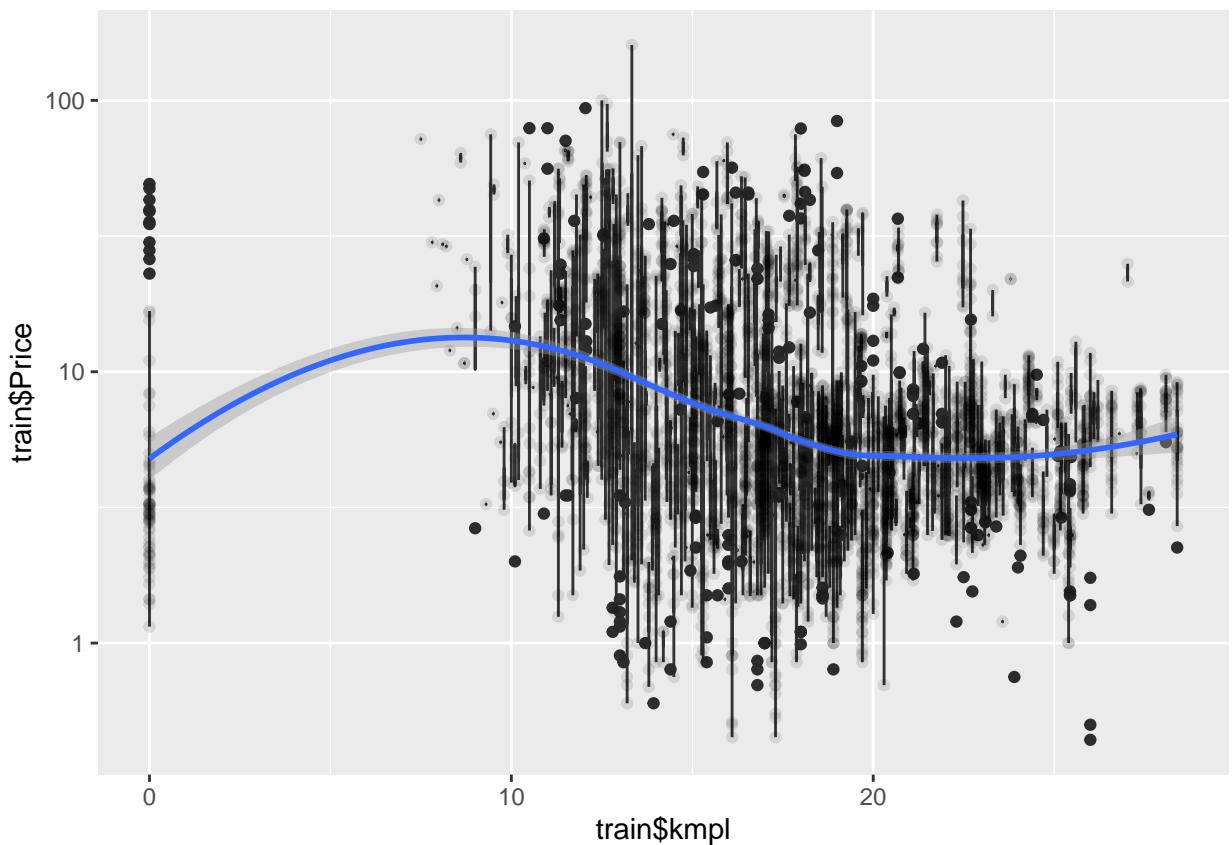
```



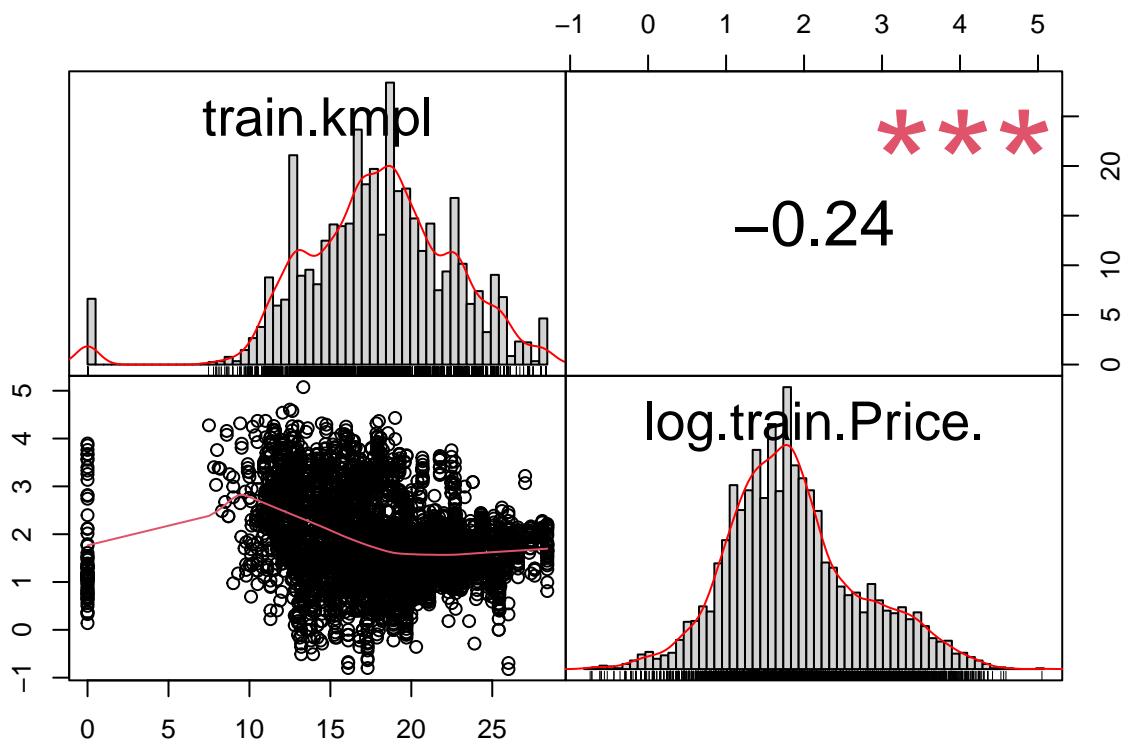
Se aprecia una buena e igualada distribución en el precio de los coches en función del número de asientos que posean. Los más caros son los biplaza debido, seguramente, a que las marcas de lujo como Lamborguini y Bently venden sobre todo deportivos biplaza. Es evidente que el número de plazas en el coche a igualdad de condiciones, afectará pero no parece altamente significativo en este dataset. También es cierto que coches biplaza sólo tenemos 11 en este dataset y es difícil sacar una conclusión con certeza.

Kmpl/Kmpkg + Price

```
ggplot(train, aes(train$kmpl, train$Price))+
  geom_boxplot(aes(group=train$kmpl))+
  geom_jitter(alpha=0.1)+
  geom_smooth(method="loess")+
  scale_y_log10()
```

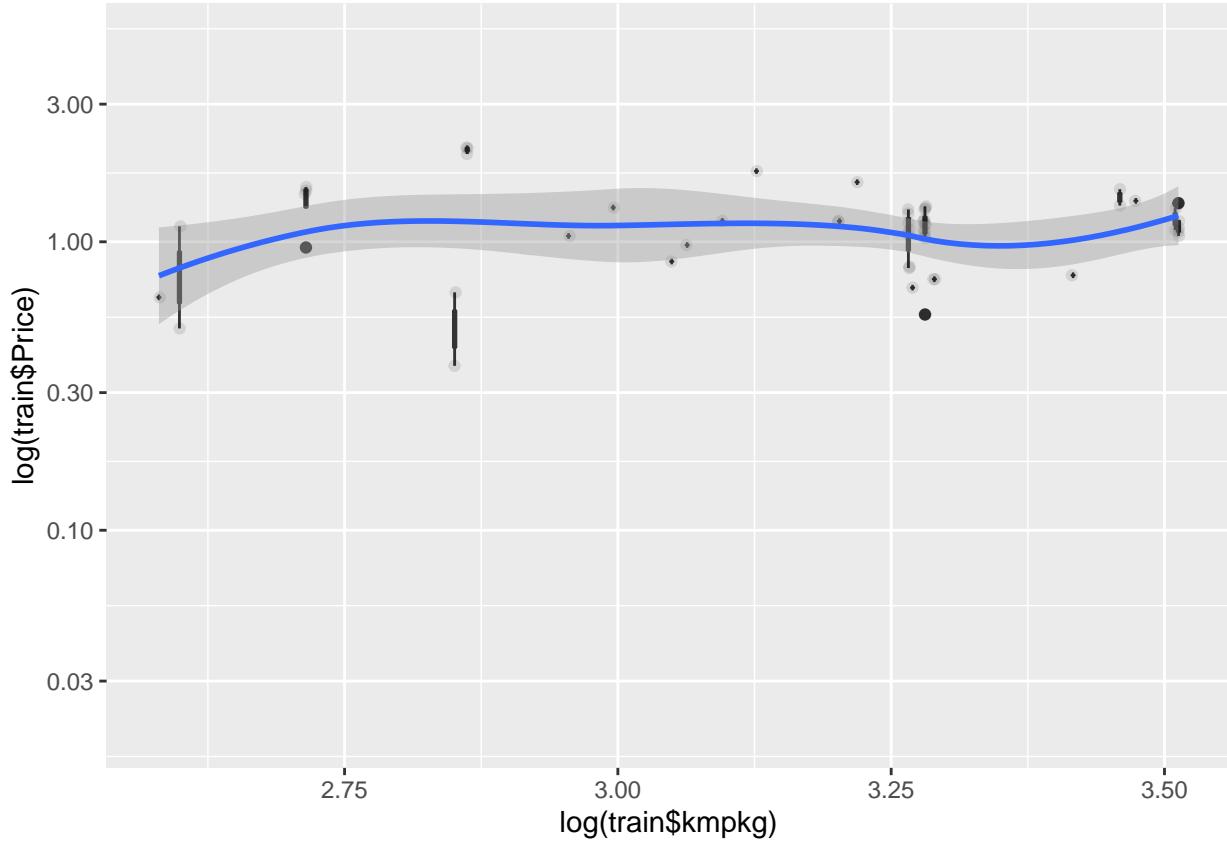


```
data.frame(train$kmpl, log(train$Price)) %>% chart.Correlation()
```

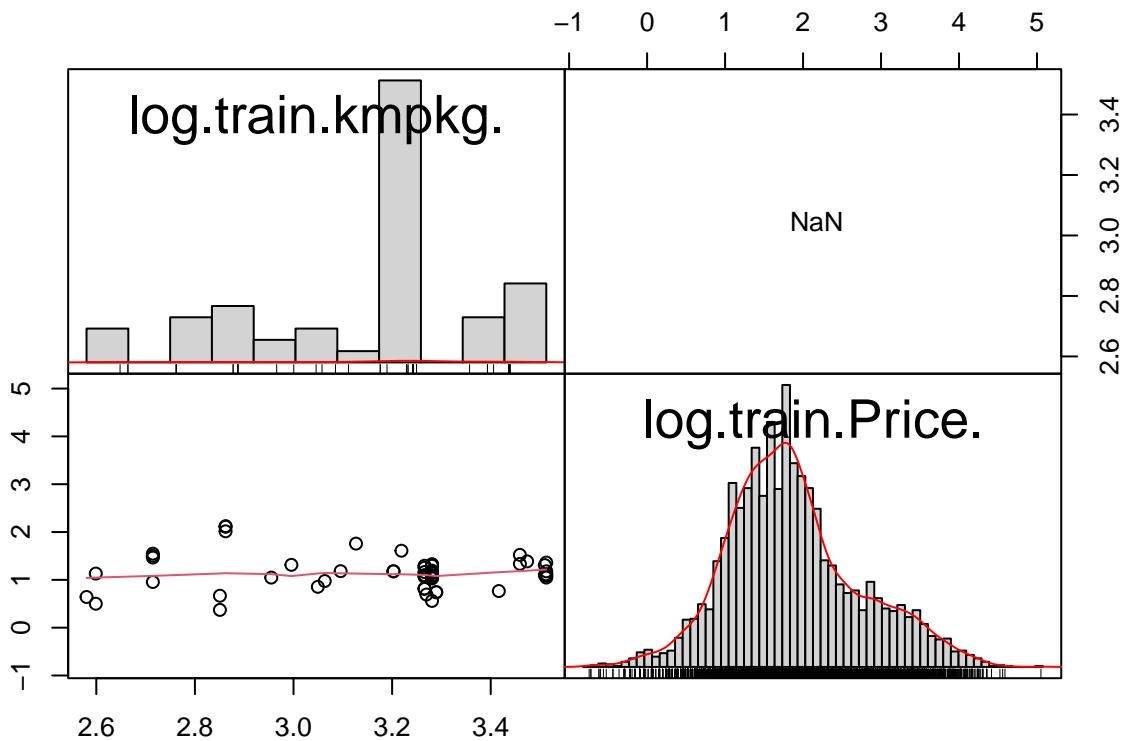


```
ggplot(train, aes(log(train$kmpl), log(train$Price)))+
  geom_boxplot(aes(group=train$kmpl))+
  geom_point(aes(group=train$kmpl))
```

```
geom_jitter(alpha=0.1)+  
geom_smooth(method="loess") +  
scale_y_log10()
```



```
data.frame(log(train$kmpkg), log(train$Price)) %>% chart.Correlation()
```

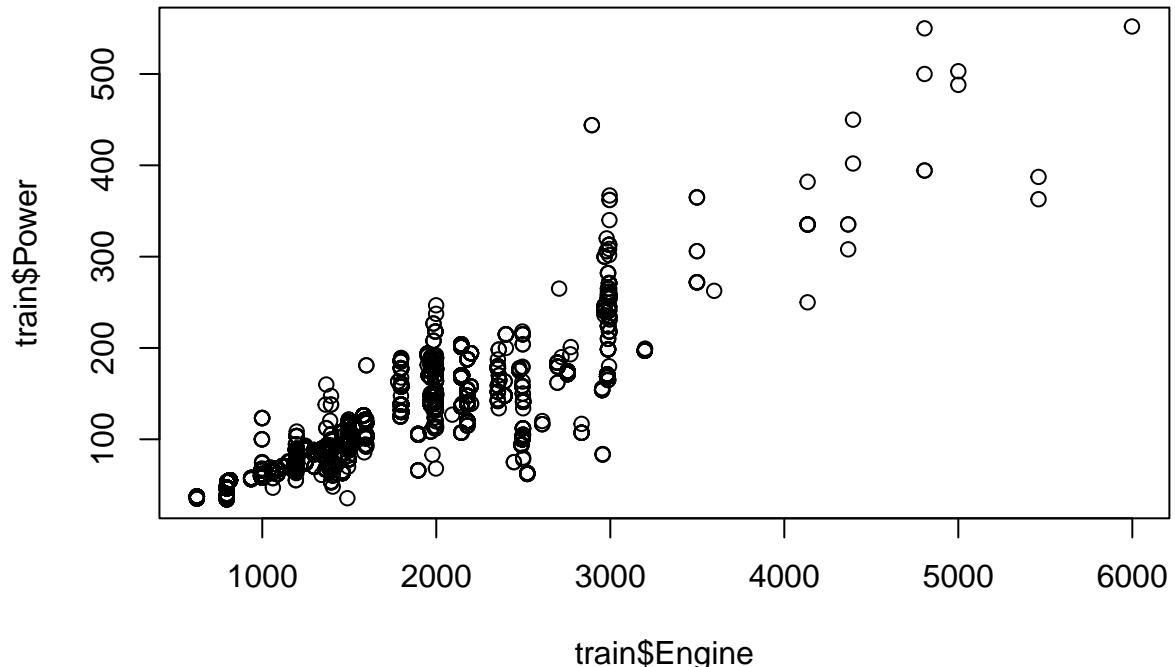


La relación entre el kilometraje y el precio no va a ser diferencial ni significativa en nuestro análisis.

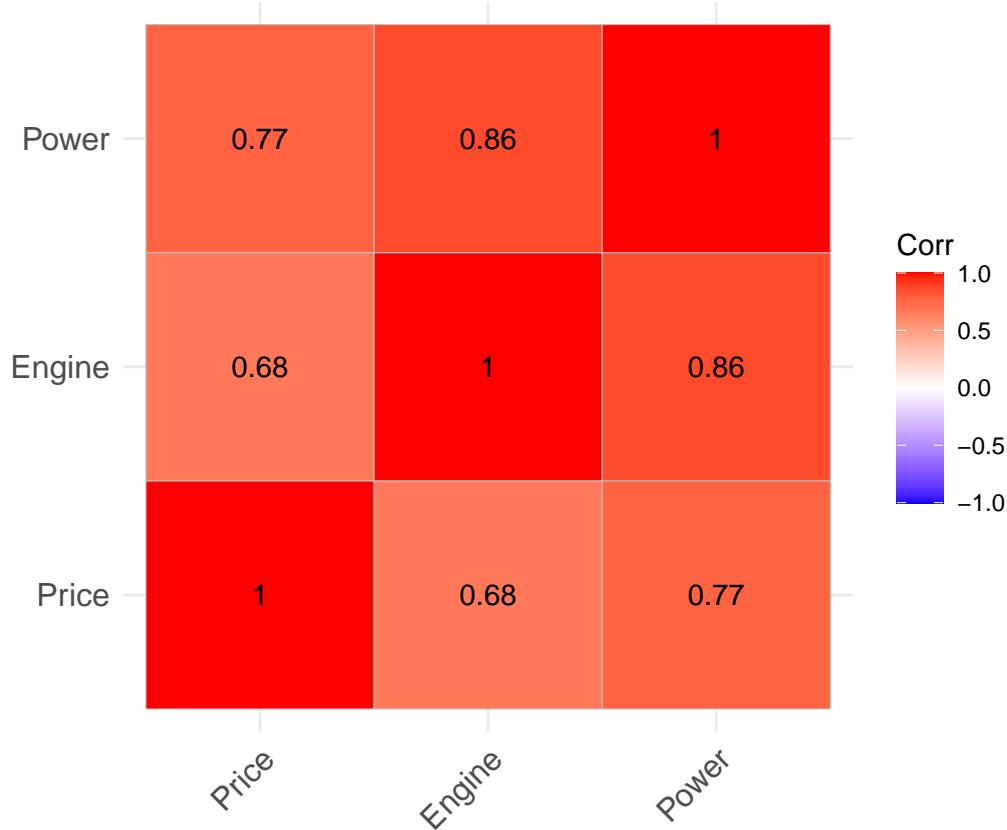
En conclusión, las variables que más afectan a nuestra variable respuesta, el precio, son: la marca, el tipo de combustible, el tipo de transmisión, la cilindrada y la potencia y en menor medida pero que también analizaremos: el año de fabricación, los dueños que ha tenido y el número de plazas que posee

Engine + Power

```
plot(train$Engine, train$Power)
```



```
train %>% select(Price, Engine, Power) %>% mutate(Price=log10(Price)) %>% cor() %>% ggcorrplot(lab=TRUE)
```



Existe una alta correlación entre las variables Engine y Power, al igual que en el mundo real, la cilindrada del motor determina la potencia de éste. Es por eso que después en el modelo nos valdrá con incluir solamente una de las dos.

AJUSTE, INTERPRETACIÓN Y DIAGNOSIS DEL MODELO DE REGRESIÓN LINEAL MÚLTIPLE

```
cor(train[, c(3, 8, 9, 11)])  
  
##          Year      Engine      Power      Price  
## Year  1.000000000 -0.06649428  0.02052167  0.3103671  
## Engine -0.06649428  1.000000000  0.86219330  0.6457693  
## Power   0.02052167  0.86219330  1.000000000  0.7622602  
## Price   0.31036712  0.64576934  0.76226015  1.0000000  
  
#Trabajos con las variables cuantitativas  
reg.fit.select=regsubsets(x=train[, c(3, 8, 9, 10)], y=train$Price)  
reg.summary=summary(reg.fit.select)  
summary(reg.fit.select)  
  
## Subset selection object  
## 4 Variables  (and intercept)  
##     Forced in Forced out
```

```

## Year      FALSE    FALSE
## Engine    FALSE    FALSE
## Power     FALSE    FALSE
## Seats     FALSE    FALSE
## 1 subsets of each size up to 4
## Selection Algorithm: exhaustive
##          Year Engine Power Seats
## 1  ( 1 ) " "   " "   "*"   " "
## 2  ( 1 ) "*"   " "   "*"   " "
## 3  ( 1 ) "*"   "*"   "*"   " "
## 4  ( 1 ) "*"   "*"   "*"   "*"

```

Vamos a proceder a seleccionar las variables con las que vamos a hacer la regresión, empezamos con las variables cuantitativas únicamente. El mejor modelo con una variable explicativa, sería el que contiene Power. El mejor modelo con dos variables explicativas, sería el que contiene Year y Power. El mejor modelo con tres, el que contiene Year, Power y Engine. El mejor modelo con cuatro variables es el que contiene las cuatro.

Estadísticos R²:

```
#Estadisticos R^2:
reg.summary$rsq
```

```
## [1] 0.5810405 0.6679395 0.6686566 0.6709138
```

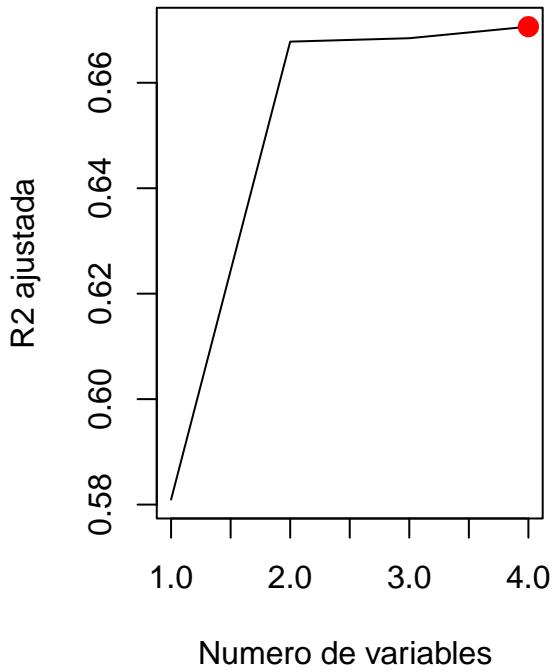
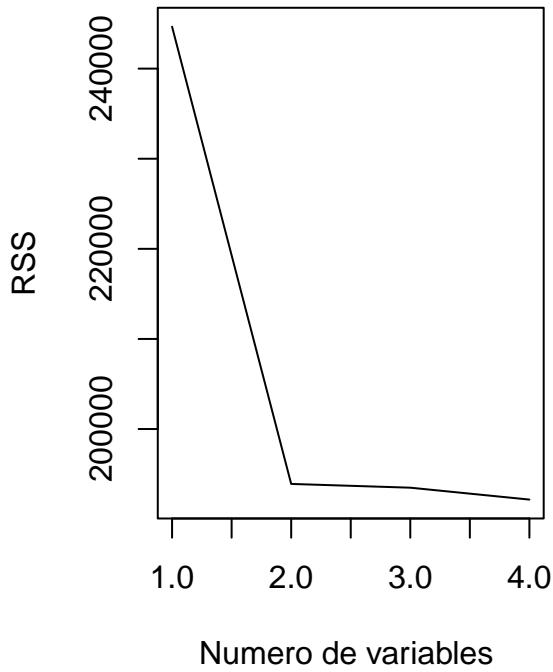
Estadísticos R² ajustado:

```
#Estadisticos R^2 ajustado:
reg.summary$adjr2
```

```
## [1] 0.5809516 0.6677985 0.6684454 0.6706341
```

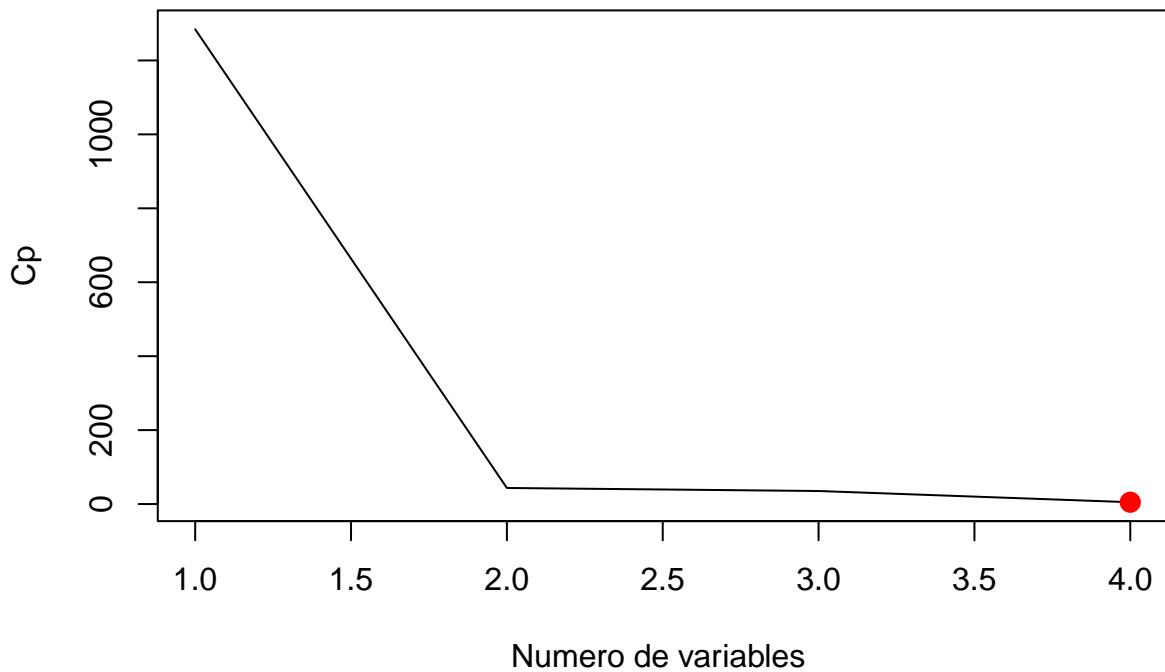
El mejor modelo es el que maximiza la R² y la R² ajustada. En este caso y teniendo en cuenta sólo 4 covariables, podríamos realizar un modelo con ellas pero siempre que se pueda simplificar y ello no reduzca los estadísticos en gran medida, puede ser práctico escoger el modelo con dos covariables.

```
par(mfrow=c(1,2))
plot(reg.summary$rss ,xlab="Número de variables ",ylab="RSS",
     type="l") #aqui usamos rss
plot(reg.summary$adjr2 ,xlab="Número de variables ",
     ylab="R2 ajustada",type="l")
num_var <- which.max(reg.summary$adjr2)
points(num_var,reg.summary$adjr2[num_var] , col="red",cex=2,pch=20)
```



CP

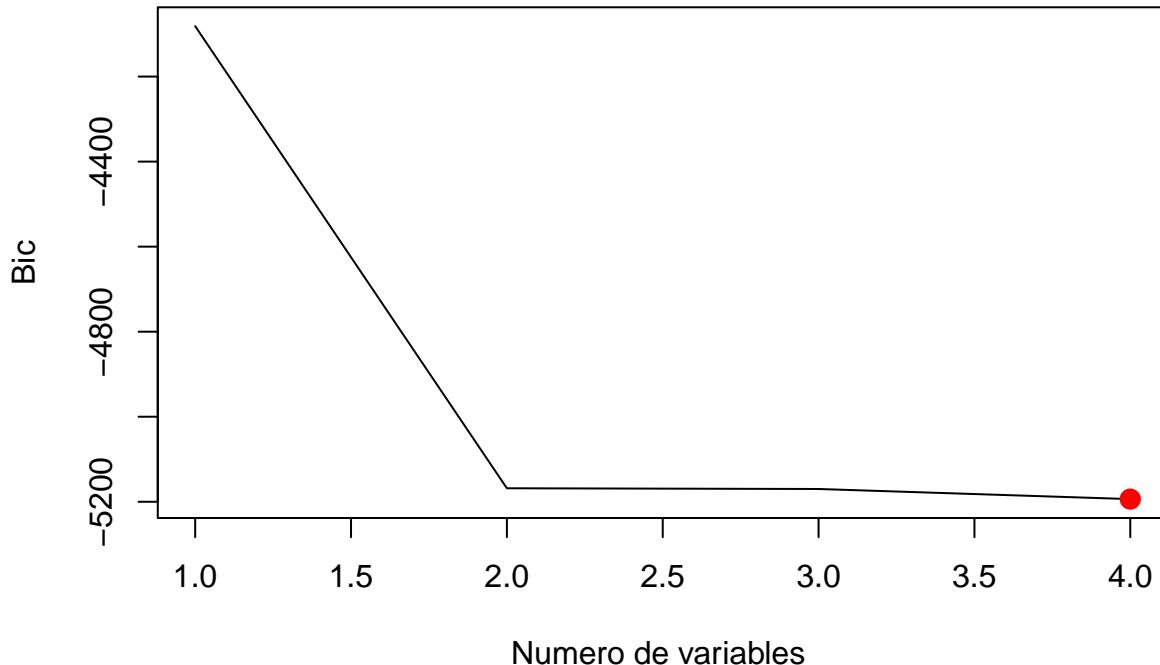
```
reg.summary$cp
plot(reg.summary$cp ,xlab="Numero de variables ",ylab="Cp",type="l")
num_var <- which.min(reg.summary$cp)
points(num_var,reg.summary$cp[num_var] , col="red",cex=2,pch=20)
```



BIC

```
reg.summary$bic
plot(reg.summary$bic ,xlab="Numero de variables ",ylab="Bic",type="l")
num_var <- which.min(reg.summary$bic)
```

```
points(num_var, reg.summary$bic[num_var], col="red", cex=2, pch=20)
```



El criterio de información de Akaike (AIC), el criterio de información de Akaike corregido (AICc) y el criterio de información bayesiano (BIC) son medidas de la calidad relativa de un modelo que representan el ajuste y el número de términos en el modelo. Una vez utilizados los criterios y estadísticos para seleccionar las variables significativas, en este caso cuantitativas, y aunque los criterios no indiquen lo mismo, tomando la iniciativa y mirando los estadísticos, se ve conveniente escoger las variables Year y Power para explicar el modelo. Así, conseguimos que se simplifique y consideraremos que se ajustan bien a los datos, además, falta aún añadir las variables cualitativas.

Ahora vamos a realizar lo mismo pero añadiendo las variables que están como factor en nuestros datos.

```
x=model.matrix(~Year+Transmission+Owner_Type+Engine+Power+Seats+Gama,data=train)
reg.fit.select=regsubsets(x[,-1], y=train$Price)
#Quitamos la intercptación que por defecto se construye en la matriz de diseño, ya que la función regsubsets
reg.summary= summary(reg.fit.select)
reg.summary

## Subset selection object
## 10 Variables (and intercept)
##          Forced in    Forced out
## Year                  FALSE     FALSE
## TransmissionManual   FALSE     FALSE
## Owner_TypeSecond     FALSE     FALSE
## Owner_TypeThird      FALSE     FALSE
## Owner_TypeFourth & Above FALSE     FALSE
## Engine                FALSE     FALSE
## Power                 FALSE     FALSE
## Seats                 FALSE     FALSE
## GamaGama baja         FALSE     FALSE
## GamaGama media        FALSE     FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
```

```

##          Year TransmissionManual Owner_TypeSecond Owner_TypeThird
## 1  ( 1 ) " "      " "           " "           " "
## 2  ( 1 ) "*"     " "           " "           " "
## 3  ( 1 ) "*"     " "           " "           " "
## 4  ( 1 ) "*"     " "           " "           " "
## 5  ( 1 ) "*"     " "           " "           " "
## 6  ( 1 ) "*"     "*"          " "           " "
## 7  ( 1 ) "*"     "*"          "*"          " "
## 8  ( 1 ) "*"     "*"          "*"          " "
##          Owner_TypeFourth & Above Engine Power Seats GamaGama baja
## 1  ( 1 ) " "      " "      "*"      " "      " "
## 2  ( 1 ) " "      " "      "*"      " "      " "
## 3  ( 1 ) " "      " "      "*"      " "      " "
## 4  ( 1 ) " "      " "      "*"      " "      "*" 
## 5  ( 1 ) " "      " "      "*"      "*"      "*" 
## 6  ( 1 ) " "      " "      "*"      "*"      "*" 
## 7  ( 1 ) " "      " "      "*"      "*"      "*" 
## 8  ( 1 ) " "      "*"      "*"      "*"      "*" 
##          GamaGama media
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) "*"
## 4  ( 1 ) "*"
## 5  ( 1 ) "*"
## 6  ( 1 ) "*"
## 7  ( 1 ) "*"
## 8  ( 1 ) "*"

```

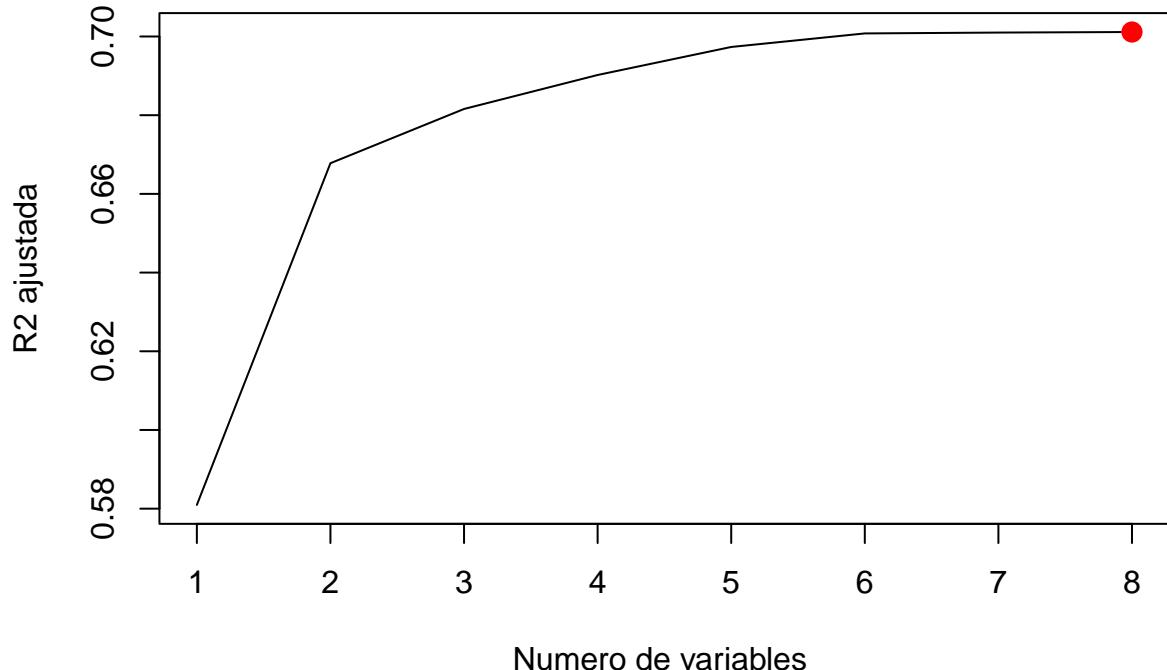
Parece ser que los coches de gama baja y gama media serán importantes en la creación de un modelo explicativo de los datos. Al intentar incluir la variable Fuel_Type nos aparece redundancia pero es una variable con importante correlación y que consideraremos al crear el modelo de regresión. Una vez añadidas las variables factor, vamos a ver que variables seleccionamos.

```
reg.summary$adjr2
```

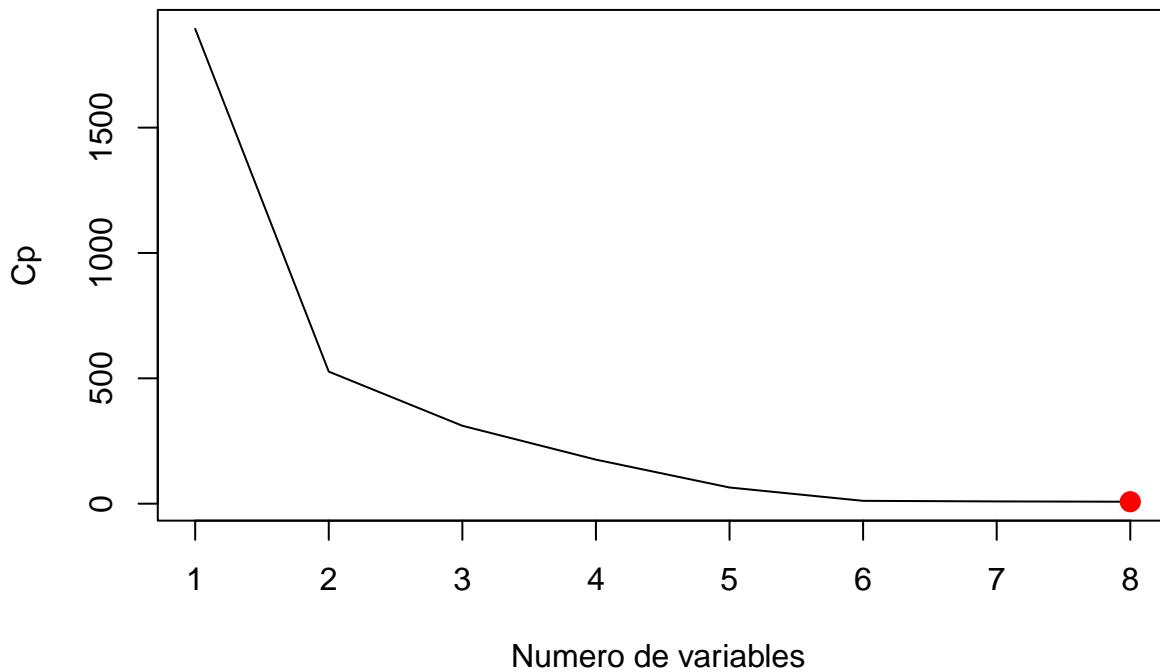
```
## [1] 0.5809516 0.6677985 0.6815782 0.6902156 0.6973410 0.7007809 0.7009960
```

```
## [8] 0.7011389
```

```
plot(reg.summary$adjr2 ,xlab="Numero de variables ", ylab="R2 ajustada",type="l")
num_var <- which.max(reg.summary$adjr2)
points(num_var,reg.summary$adjr2[num_var] , col="red",cex=2,pch=20)
```



```
reg.summary$cp
plot(reg.summary$cp ,xlab="Numero de variables ",ylab="Cp",type="l")
var <- which.min(reg.summary$cp)
points(var,reg.summary$cp[var] , col="red",cex=2,pch=20)
```

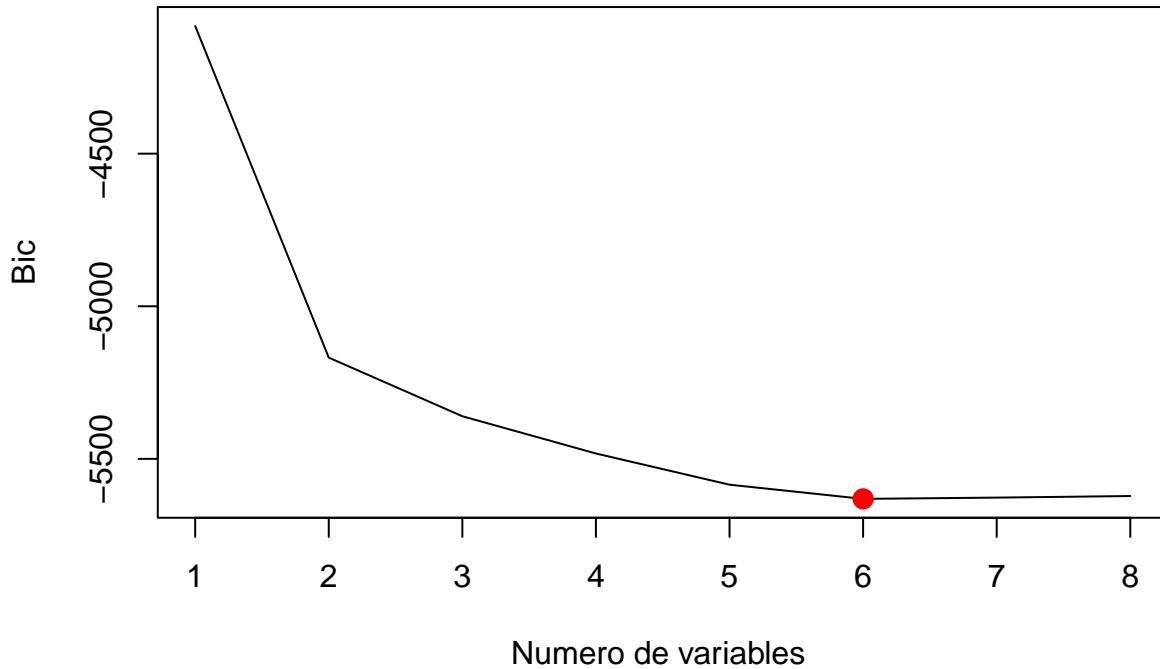


```
reg.summary$bic
## [1] -4081.566 -5168.213 -5360.338 -5482.435 -5584.603 -5630.996 -5626.927
## [8] -5621.724
```

```

plot(reg.summary$bic ,xlab="Numero de variables ",ylab="Bic",type="l")
var <- which.min(reg.summary$bic)
points(var,reg.summary$bic[var], col="red",cex=2,pch=20)

```



Gracias a los métodos para saber cual es el mejor modelo y al summary se obtiene que el mejor modelo de regresión lineal multiple es el que contiene: Year, Transmission manual, OwnerType second, Engine, Power, Seats, Gama baja y Gama media y con 6 covariables sería el modelo formado por: Year, Transmission manual, Power, Seats, Gama baja y gama media. Teniendo en cuenta la variable Fuel_Type también, vamos a añadirla al modelo. Estos métodos tienen muy en cuenta el valor de la R adjusted pero nosotros lo evaluaremos por nosotros mismos y decidiremos cuánto de importante es el aumento de ese estadístico a la hora de aumentar variables en el modelo.

Para ajustar un modelo de regresión lineal múltiple mediante mínimos cuadrados, se utiliza la función lm(). Mediante esta función ajustamos una recta de regresión, con Price como variable respuesta. Los posibles modelos que podemos utilizar para explicar nuestros datos, teniendo en cuenta el análisis anterior, son:

```

mod0=lm(log10(Price) ~ Year+Power,data=train) #2 variables
mod1=lm(log10(Price) ~ Year+Fuel_Type+Power,data=train) #2 variables +fuel_type
mod2=lm(log10(Price) ~ Year+Power+Gama,data=train) #3variables
mod3=lm(log10(Price) ~ Year+Fuel_Type+Power+Gama,data=train) #3 variables +fuel
mod4=lm(log10(Price) ~ Year+Power+Seats+Gama,data=train) #4 variables
mod5=lm(log10(Price) ~ Year+Fuel_Type+Power+Seats+Gama,data=train) #4 var + fuel_type
mod6=lm(log10(Price) ~ Year+ Transmission+Power+Seats+Gama,data=train) #5 var
mod7=lm(log10(Price) ~ Year+Fuel_Type+Transmission+Power+Seats+Gama,data=train) #5 var + fuel
mod8=lm(log10(Price) ~ Year+Transmission+Owner_Type+Power+Seats+Gama,data=train) #6var
mod9=lm(log10(Price) ~ Year+Fuel_Type+Transmission+Owner_Type+Power+Seats+Gama,data=train) #6var + fuel
mod10=lm(log10(Price) ~ Year+Transmission+Owner_Type+Power+Engine+Seats+Gama,data=train) #7var
mod11=lm(log10(Price) ~ Year+Fuel_Type+Transmission+Owner_Type+Power+Engine+Seats+Gama,data=train) #7va

```

Vamos a chequearlos para ver cual obtiene mejores estadísticos:

```
compare_performance(mod0,mod1,mod2,mod3,mod4,mod5,mod6,mod7,mod8,mod9,mod10,mod11, rank=TRUE)
```

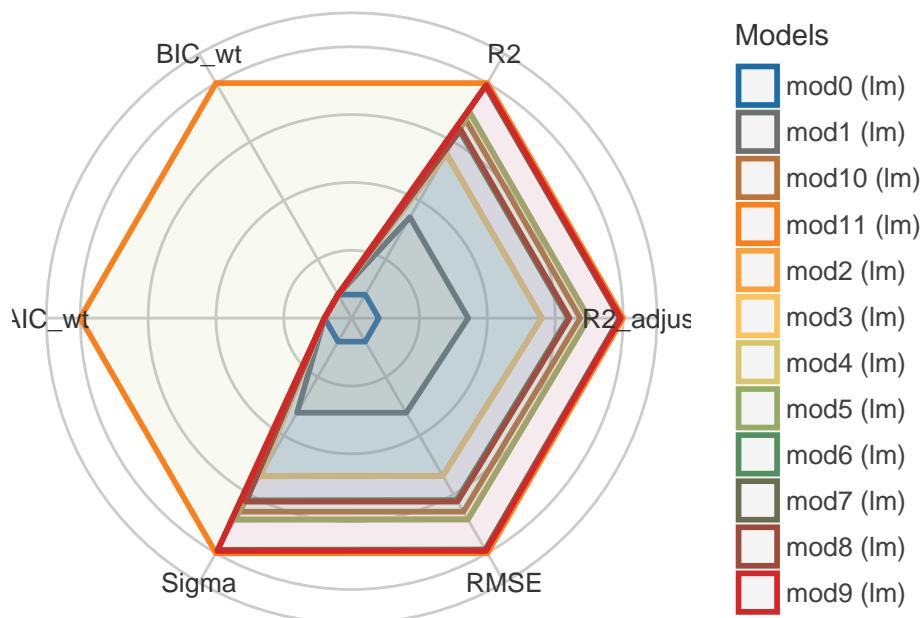
```
## # Comparison of Model Performance Indices
```

```

## 
## Name | Model | R2 | R2 (adj.) | RMSE | Sigma | AIC weights | BIC weights | Performance-Score
## -----
## mod11 | lm | 0.892 | 0.892 | 0.124 | 0.124 | 1.000 | 1.000 | 100.00%
## mod9 | lm | 0.891 | 0.891 | 0.124 | 0.124 | < 0.001 | < 0.001 | 66.01%
## mod7 | lm | 0.891 | 0.891 | 0.124 | 0.125 | < 0.001 | < 0.001 | 65.62%
## mod5 | lm | 0.881 | 0.881 | 0.130 | 0.130 | < 0.001 | < 0.001 | 56.71%
## mod3 | lm | 0.881 | 0.881 | 0.130 | 0.130 | < 0.001 | < 0.001 | 56.49%
## mod10 | lm | 0.878 | 0.878 | 0.131 | 0.131 | < 0.001 | < 0.001 | 54.29%
## mod8 | lm | 0.875 | 0.875 | 0.133 | 0.133 | < 0.001 | < 0.001 | 51.24%
## mod6 | lm | 0.875 | 0.874 | 0.133 | 0.134 | < 0.001 | < 0.001 | 50.78%
## mod4 | lm | 0.866 | 0.866 | 0.138 | 0.138 | < 0.001 | < 0.001 | 43.44%
## mod2 | lm | 0.866 | 0.866 | 0.138 | 0.138 | < 0.001 | < 0.001 | 43.37%
## mod1 | lm | 0.843 | 0.843 | 0.149 | 0.149 | < 0.001 | < 0.001 | 23.40%
## mod0 | lm | 0.815 | 0.815 | 0.162 | 0.162 | < 0.001 | < 0.001 | 0.00%
plot(compare_performance(mod0,mod1,mod2,mod3,mod4,mod5,mod6,mod7,mod8,mod9,mod10,mod11, rank=TRUE))

```

Comparison of Model Indices



Analizando cada uno de los modelos, sus índices y en qué medida afecta el aumento o disminución de variables, se decide que los mejores modelos hasta el momento son:

- El modelo nº 2 tiene como variables: Year, Power y Gama
- El modelo nº 3 tiene como variables: Year, Fuel_Type, Power y Gama
- El modelo nº 6 tiene como variables: Year, Transmission, Power, Seats y Gama
- El modelo nº 7 tiene como variables: Year,Fuel_Type,Transmission,Power,Seats y Gama
- El modelo nº 11 tiene como variables: Year, Fuel_Type, Transmission, Owner_Type, Power, Engine, Seats y Gama

```
compare_performance(mod2,mod3,mod6,mod7,mod11, rank=TRUE)
```

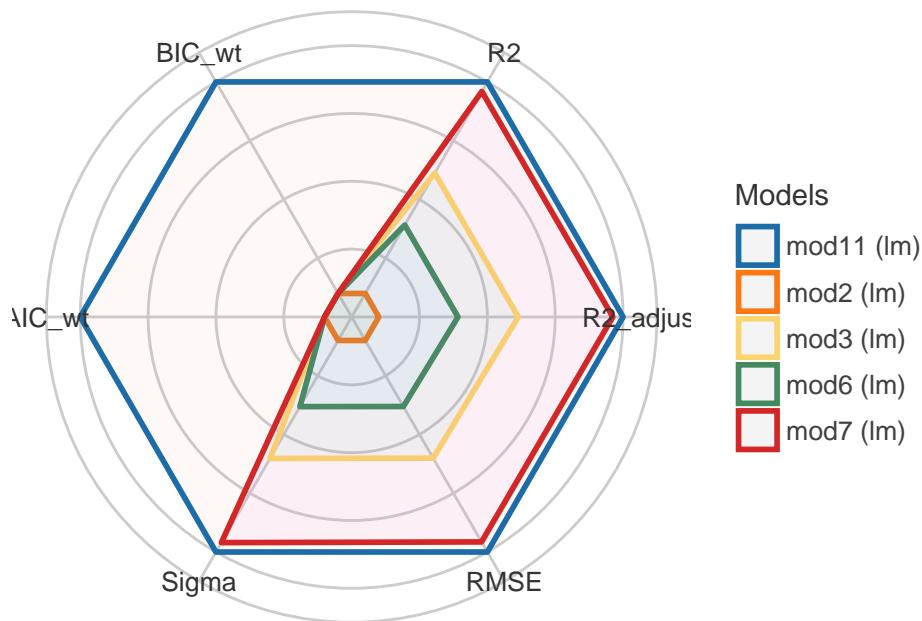
```
## # Comparison of Model Performance Indices
```

```

## 
## Name | Model | R2 | R2 (adj.) | RMSE | Sigma | AIC weights | BIC weights | Performance-Score
## -----
## mod11 | lm | 0.892 | 0.892 | 0.124 | 0.124 | 1.000 | 1.000 | 100.00%
## mod7 | lm | 0.891 | 0.891 | 0.124 | 0.125 | < 0.001 | < 0.001 | 63.68%
## mod3 | lm | 0.881 | 0.881 | 0.130 | 0.130 | < 0.001 | < 0.001 | 37.57%
## mod6 | lm | 0.875 | 0.874 | 0.133 | 0.134 | < 0.001 | < 0.001 | 21.21%
## mod2 | lm | 0.866 | 0.866 | 0.138 | 0.138 | < 0.001 | < 0.001 | 0.00%
plot(compare_performance(mod2,mod3,mod6,mod7,mod11, rank=TRUE))

```

Comparison of Model Indices



Se va a analizar primero el modelo con más variables, el **modelo 11**:

```
summary(mod11)
```

```

## 
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + Transmission +
##     Owner_Type + Power + Engine + Seats + Gama, data = train)
## 
## Residuals:
##      Min        1Q    Median        3Q       Max
## -1.68108 -0.07368  0.00746  0.08011  0.61876
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.063e+02  1.291e+00 -82.342 < 2e-16 ***
## Year         5.305e-02  6.407e-04  82.806 < 2e-16 ***
## Fuel_TypeDiesel 1.108e-01  1.865e-02   5.943 3.00e-09 ***
## Fuel_TypeLPG   4.508e-02  4.757e-02   0.948 0.343281
## Fuel_TypePetrol 8.686e-03  1.854e-02   0.469 0.639391
## TransmissionManual -1.148e-01  5.512e-03 -20.823 < 2e-16 ***

```

```

## Owner_TypeSecond      -1.730e-02  5.247e-03  -3.297  0.000986 ***
## Owner_TypeThird       -4.270e-02  1.388e-02  -3.077  0.002104 **
## Owner_TypeFourth & Above 6.480e-02  5.088e-02   1.274  0.202869
## Power                 2.984e-03  8.766e-05  34.043  < 2e-16 ***
## Engine                4.687e-05  8.436e-06   5.556  2.92e-08 ***
## Seats                 -3.919e-03  3.015e-03  -1.300  0.193741
## GamaGama baja        -3.108e-01  9.507e-03 -32.687  < 2e-16 ***
## GamaGama media        -1.552e-01  6.295e-03 -24.648  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.124 on 4697 degrees of freedom
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8916
## F-statistic:  2982 on 13 and 4697 DF,  p-value: < 2.2e-16

```

Si observamos los p-valores de los test de hipótesis para los coeficientes individuales, observamos que para alguna variable no se puede rechazar que el coeficiente sea cero. Eliminamos por tanto la variable "Seats" de nuestro modelo 11.

```
mod11=lm(log10(Price) ~ Year+Fuel_Type+Transmission+Owner_Type+Power+Engine+Gama,data=train)
summary(mod11)
```

```

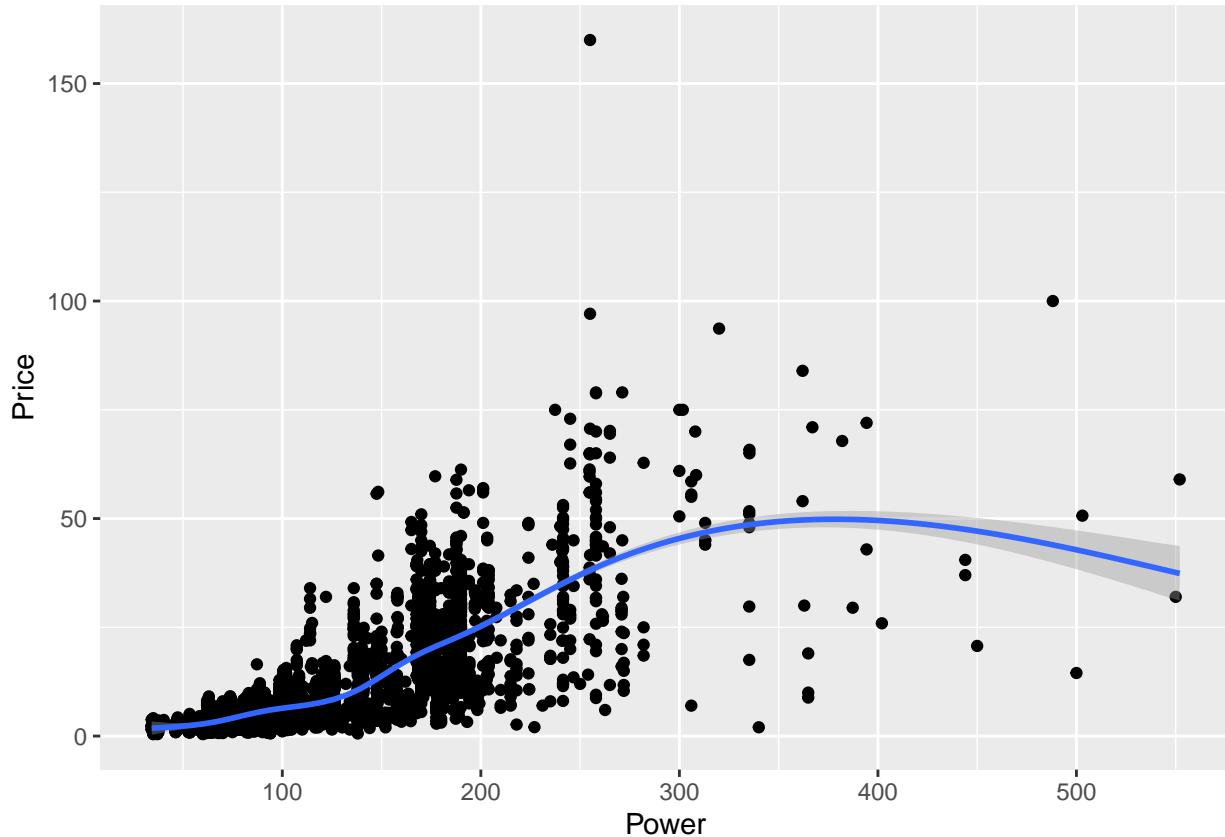
##
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + Transmission +
##     Owner_Type + Power + Engine + Gama, data = train)
##
## Residuals:
##      Min        1Q        Median         3Q        Max
## -1.68299 -0.07361  0.00763  0.08053  0.62390
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -1.061e+02  1.282e+00 -82.763  < 2e-16 ***
## Year                    5.295e-02  6.354e-04  83.326  < 2e-16 ***
## Fuel_TypeDiesel          1.107e-01  1.865e-02   5.935 3.15e-09 ***
## Fuel_TypeLPG              4.494e-02  4.757e-02   0.945  0.34483
## Fuel_TypePetrol           8.635e-03  1.854e-02   0.466  0.64140
## TransmissionManual      -1.161e-01  5.421e-03 -21.412  < 2e-16 ***
## Owner_TypeSecond          -1.724e-02  5.247e-03  -3.286  0.00102 **
## Owner_TypeThird            -4.301e-02  1.388e-02  -3.099  0.00195 **
## Owner_TypeFourth & Above  6.327e-02  5.087e-02   1.244  0.21363
## Power                     3.036e-03  7.799e-05  38.934  < 2e-16 ***
## Engine                     4.107e-05  7.159e-06   5.736 1.03e-08 ***
## GamaGama baja              -3.093e-01  9.441e-03 -32.760  < 2e-16 ***
## GamaGama media              -1.535e-01  6.163e-03 -24.905  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.124 on 4698 degrees of freedom
## Multiple R-squared:  0.8919, Adjusted R-squared:  0.8916
## F-statistic:  3230 on 12 and 4698 DF,  p-value: < 2.2e-16

```

En algunos casos, para que la relación entre la variable respuesta y los regresores sea lineal, éstos requerirán de una transformación. Por ejemplo, observemos la relación entre la variable Price y Power:

```
train %>% ggplot(aes(x= Power, y=Price)) +
  geom_point()+
  geom_smooth()
```

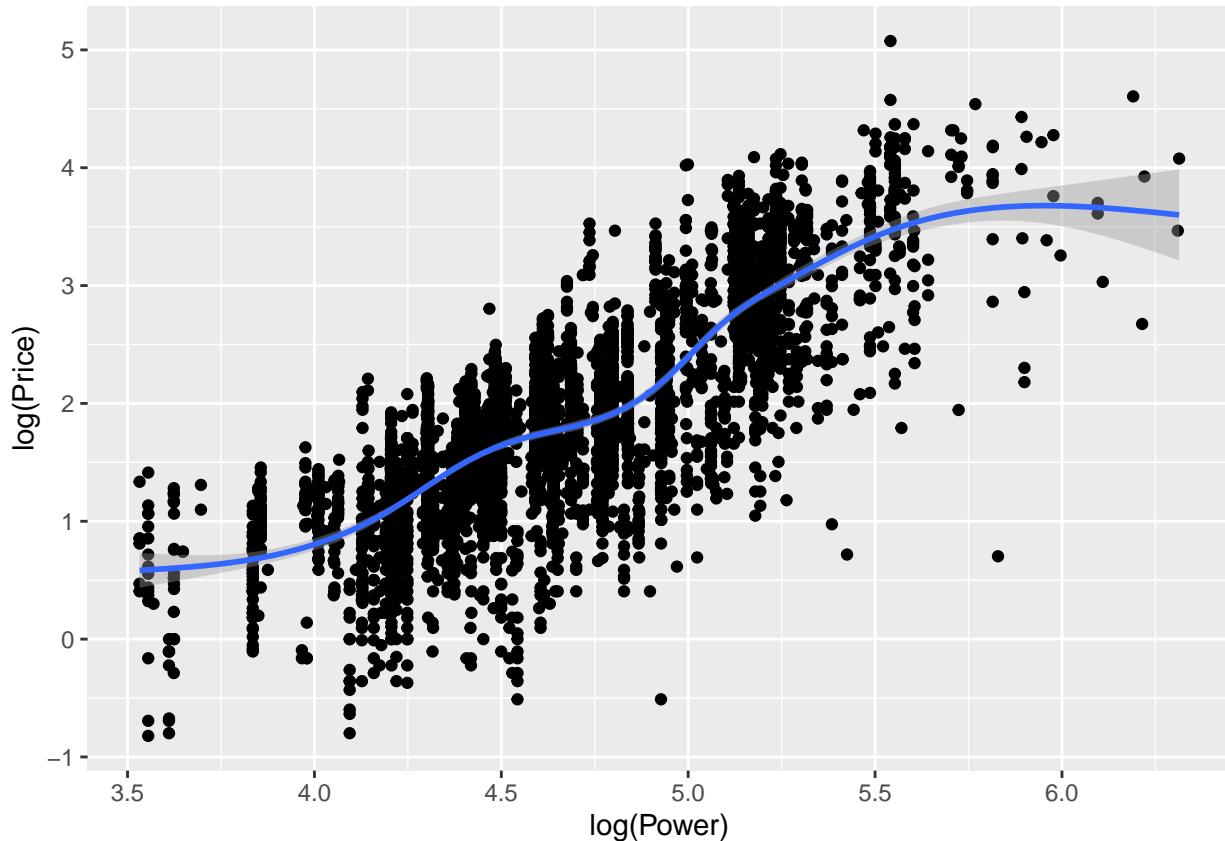
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Se puede observar, que claramente no existe una relación lineal entre ambas. Pero si aplicamos una transformación no lineal, la relación entre ambas $\log(\text{Power})$ y $\log(\text{Price})$ pasa a ser lineal aproximadamente.

```
train %>% ggplot(aes(x= log(Power), y=log(Price))) +
  geom_point()+
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



```
mod11=lm(log10(Price) ~ Year+Fuel_Type+Transmission+Owner_Type+log(Power)+Engine+Gama, data=train)
summary(mod11)
```

```
##
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + Transmission +
##     Owner_Type + log(Power) + Engine + Gama, data = train)
##
## Residuals:
##      Min        1Q        Median        3Q       Max
## -1.48338 -0.07187  0.00402  0.07936  0.56930
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -1.052e+02  1.249e+00 -84.245 < 2e-16 ***
## Year                     5.179e-02  6.214e-04  83.347 < 2e-16 ***
## Fuel_TypeDiesel          4.437e-02  1.822e-02   2.436  0.01490 *
## Fuel_TypeLPG              4.703e-02  4.631e-02   1.016  0.30990
## Fuel_TypePetrol           -3.786e-02  1.812e-02  -2.090  0.03670 *
## TransmissionManual      -1.168e-01  5.194e-03 -22.487 < 2e-16 ***
## Owner_TypeSecond          -2.185e-02  5.109e-03  -4.276 1.94e-05 ***
## Owner_TypeThird            -4.099e-02  1.351e-02  -3.033  0.00243 **
## Owner_TypeFourth & Above  3.751e-02  4.951e-02   0.758  0.44877
## log(Power)                 4.012e-01  9.309e-03  43.100 < 2e-16 ***
## Engine                    4.608e-05  6.614e-06   6.968 3.66e-12 ***
## GamaGama baja             -2.858e-01  9.222e-03 -30.989 < 2e-16 ***
## GamaGama media            -1.537e-01  6.000e-03 -25.622 < 2e-16 ***
```

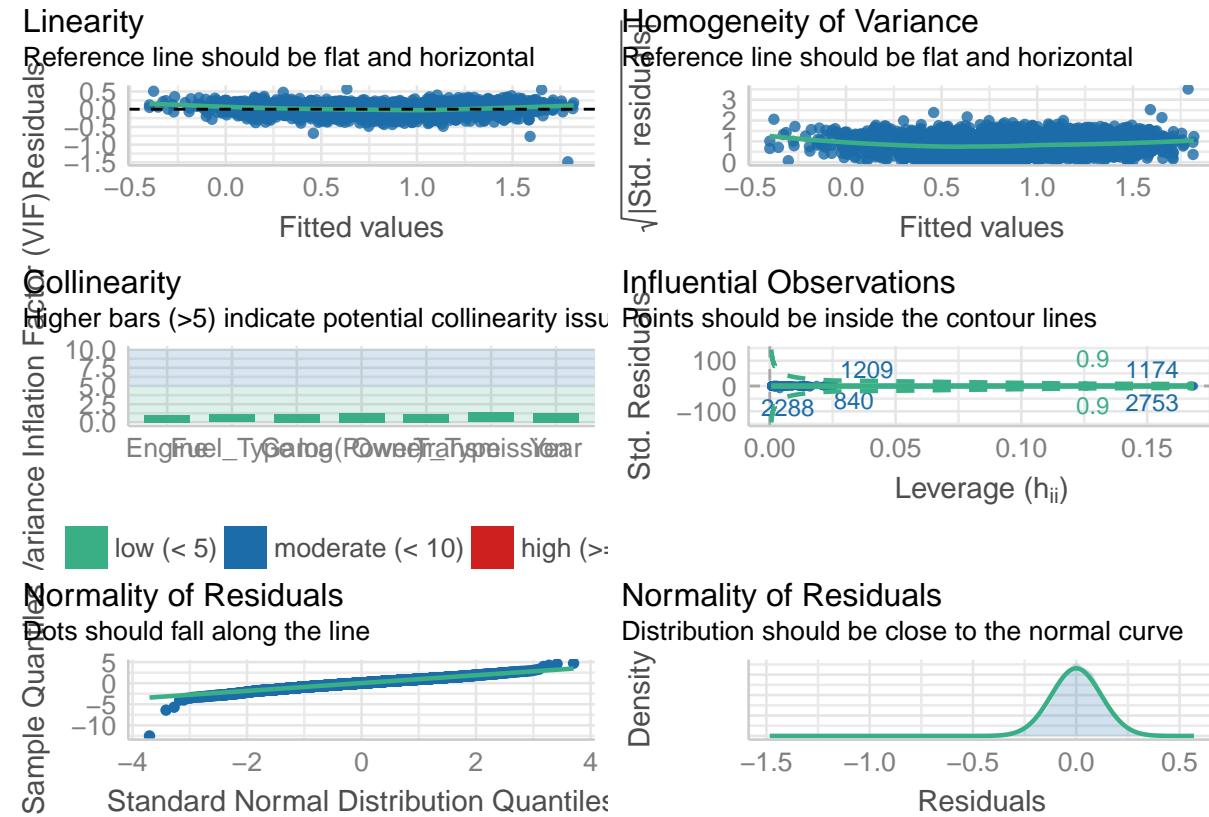
```

## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1207 on 4698 degrees of freedom
## Multiple R-squared: 0.8975, Adjusted R-squared: 0.8973
## F-statistic: 3429 on 12 and 4698 DF, p-value: < 2.2e-16

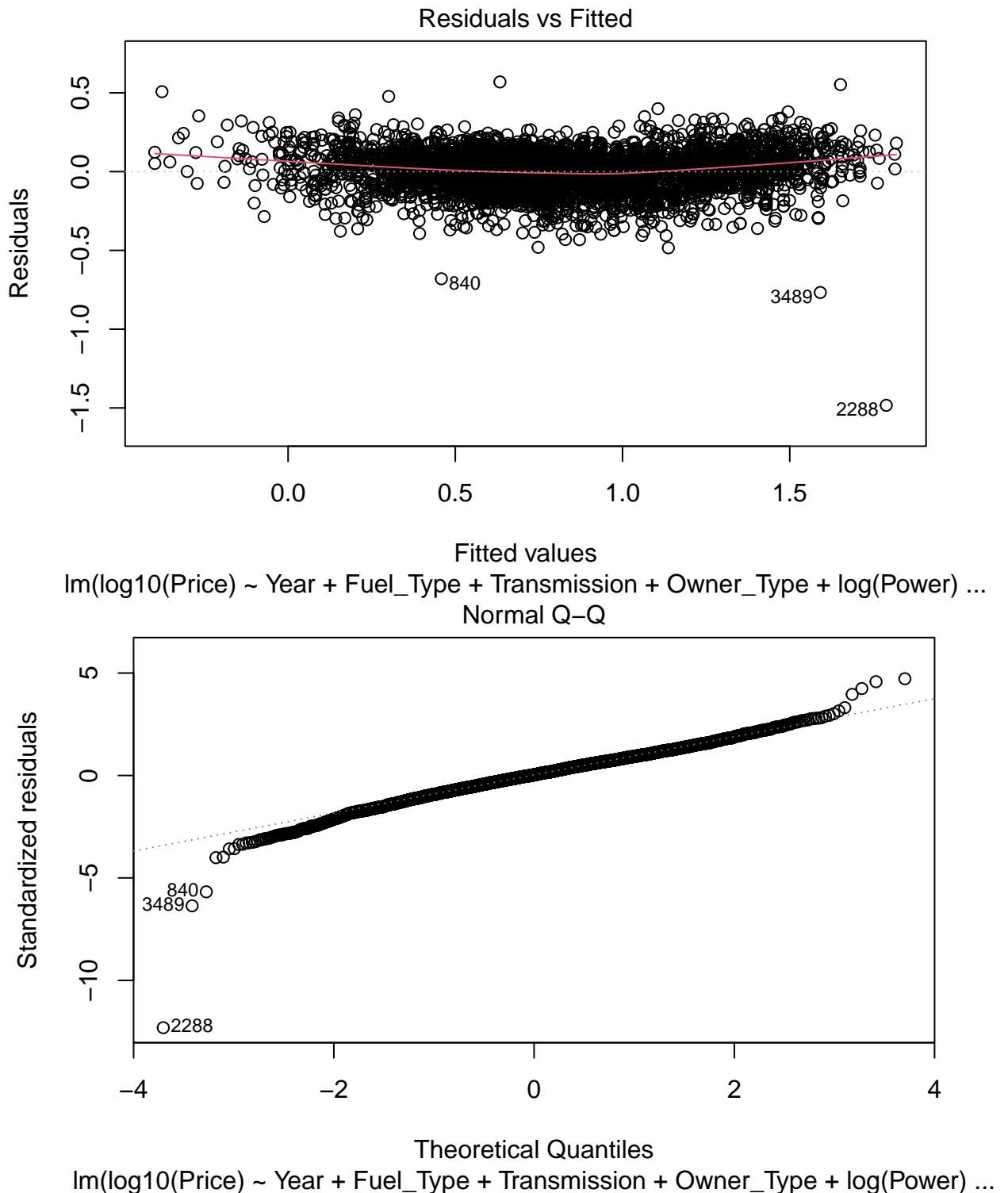
```

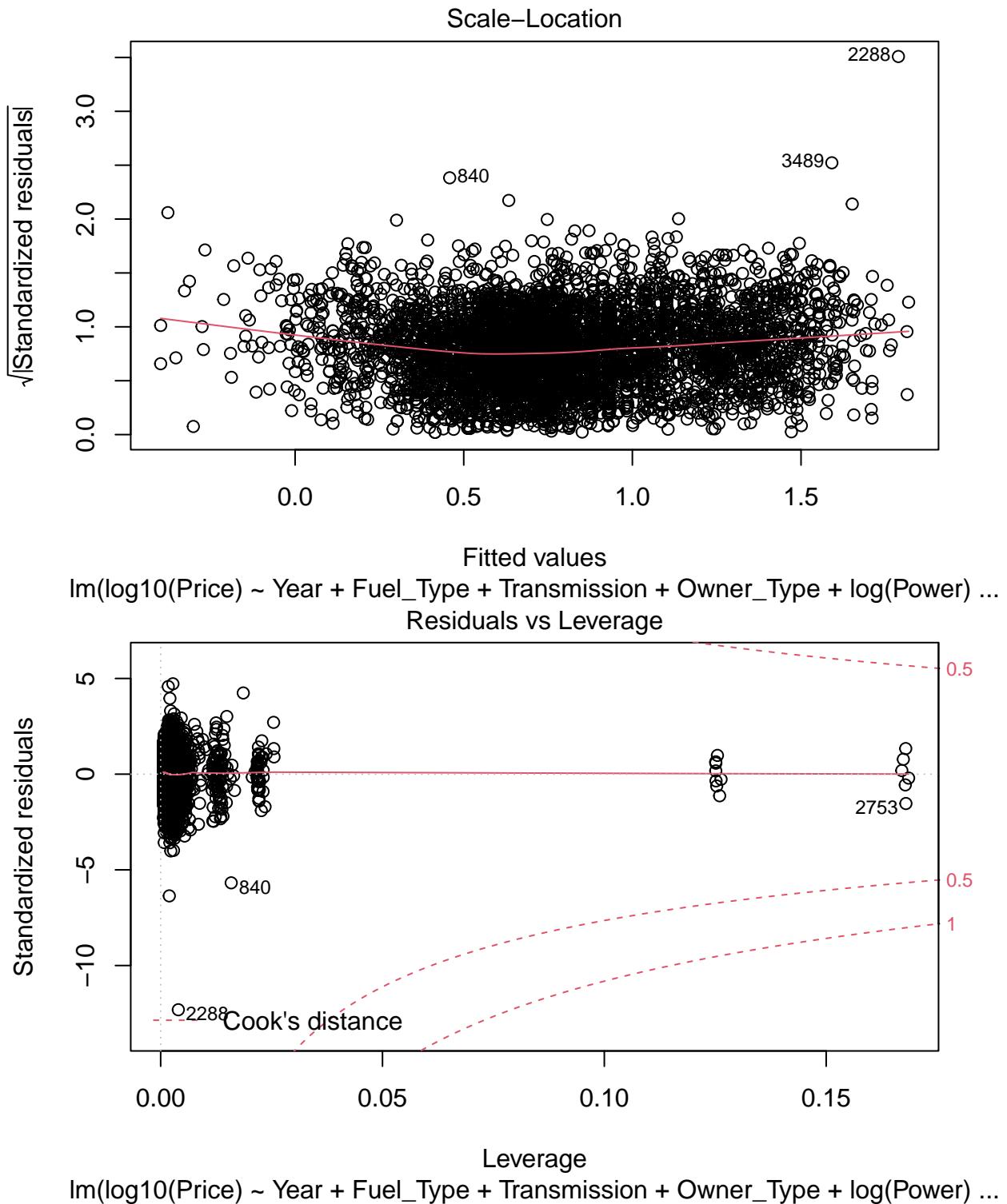
Después de construir el modelo de regresión, con la variable transformada, se observa que el valor del coeficiente relacionado con la variable Power es mucho mayor. Además, el ajuste mejora pasando de un $R^2_{adj}=0.891$ a $R^2_{adj}=0.897$. Haciendo una diagnóstico del modelo:

```
check_model(mod11)
```



```
plot(mod11)
```





En el gráfico de *Residuos vs Ajustes* se observa que la media de los residuos es prácticamente cero, luego la linealidad del modelo no se viola.

La varianza de los residuos es constante por lo que tampoco se viola la homocedasticidad del modelo, el ajuste es bueno. Lo que si se observa es la aparición de puntos candidatos a ser outliers.

Con el *Q-Q Plot* vemos que los residuos siguen una distribución normal excepto en las colas que varía un

poco. Por tanto no se puede asumir que los estimadores de los coeficientes tengan una distribución normal. En el gráfico de *Residuals vs Leverage* observamos que hay puntos con alto leverage pero ninguno cae fuera de los límites de la distancia de Cook.

```
dwtest(mod11, alternative = "two.sided")

##
## Durbin-Watson test
##
## data: mod11
## DW = 1.9902, p-value = 0.7383
## alternative hypothesis: true autocorrelation is not 0

vif(mod11)

##          GVIF Df GVIF^(1/(2*Df))
## Year      1.272090  1      1.127870
## Fuel_Type 1.350666  3      1.051376
## Transmission 1.781488  1      1.334724
## Owner_Type 1.196547  3      1.030358
## log(Power) 4.875529  1      2.208060
## Engine     5.112816  1      2.261154
## Gama       2.412373  2      1.246267
```

Además, gracias al *Durbin Watson test* sabemos que no hay evidencias de autocorrelación. Por otro lado, tenemos un problema de multicolinealidad cuando las variables explicativas (regresores) están relacionadas entre sí, de forma que solapan información para predecir la respuesta. Cuando tenemos multicolinealidad, la estimación de los coeficientes del modelo no es muy estable y por tanto los errores de estimación son grandes. Utilizamos el VIF para comprobarlo. Los límites de referencia que se suelen emplear son:

- VIF = 1: Ausencia total de colinealidad.
- 1 < VIF < 5: La regresión puede verse afectada por cierta colinealidad.
- 5 < VIF < 10: Causa de preocupación.

y vemos que no tenemos valores elevados, por tanto no hay problemas de multicolinealidad. También lo podemos comprobar gracias al gráfico “Collinearity”. Aunque entre los distintos tipos de gasolina si que existe cierta colinealidad, ello no afectará al modelo. Parece que no hay violaciones a los supuestos de OLS. Nuestro modelo parece decente, con un R² adj de aproximadamente el 90%, realmente bueno ya para un modelo lineal. OLS son las distancias verticales entre las respuestas observadas en la muestra y las respuestas del modelo. Como conclusión a este modelo diríamos que se adecúa a los datos, pero veamos si reduciendo el número de variables, es decir, simplificando el problema, podemos mantener la buena explicatividad que necesitamos.

Se va a analizar ahora el siguiente modelo, el **modelo 7**:

```
summary(mod7)

##
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + Transmission +
##     Power + Seats + Gama, data = train)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -1.70018 -0.07279  0.00669  0.08163  0.65734
##
## Coefficients:
```

```

##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -1.066e+02  1.165e+00 -91.539 < 2e-16 ***
## Year                  5.321e-02  5.779e-04  92.073 < 2e-16 ***
## Fuel_TypeDiesel       1.160e-01  1.872e-02   6.199 6.18e-10 ***
## Fuel_TypeLPG          4.337e-02  4.779e-02   0.907  0.3643
## Fuel_TypePetrol        7.541e-03  1.863e-02   0.405  0.6856
## TransmissionManual   -1.127e-01  5.530e-03 -20.386 < 2e-16 ***
## Power                 3.375e-03  5.252e-05  64.260 < 2e-16 ***
## Seats                  4.962e-03  2.568e-03   1.932  0.0534 .
## GamaGama baja         -3.187e-01  9.460e-03 -33.694 < 2e-16 ***
## GamaGama media        -1.616e-01  6.191e-03 -26.107 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1246 on 4701 degrees of freedom
## Multiple R-squared:  0.8908, Adjusted R-squared:  0.8906
## F-statistic:  4260 on 9 and 4701 DF,  p-value: < 2.2e-16

```

Si observamos los p-valores de los test de hipótesis para los coeficientes individuales, observamos que para alguna variable no se puede rechazar que el coeficiente sea cero. Eliminamos por tanto la variable “Seats” de nuestro modelo 7. Además, gracias al análisis de antes, podemos meter Power transformada como logaritmo para permitirle esa linealidad con el precio.

```
mod7=lm(log10(Price) ~ Year+Fuel_Type+Transmission+log(Power)+Gama,data=train)
summary(mod7)
```

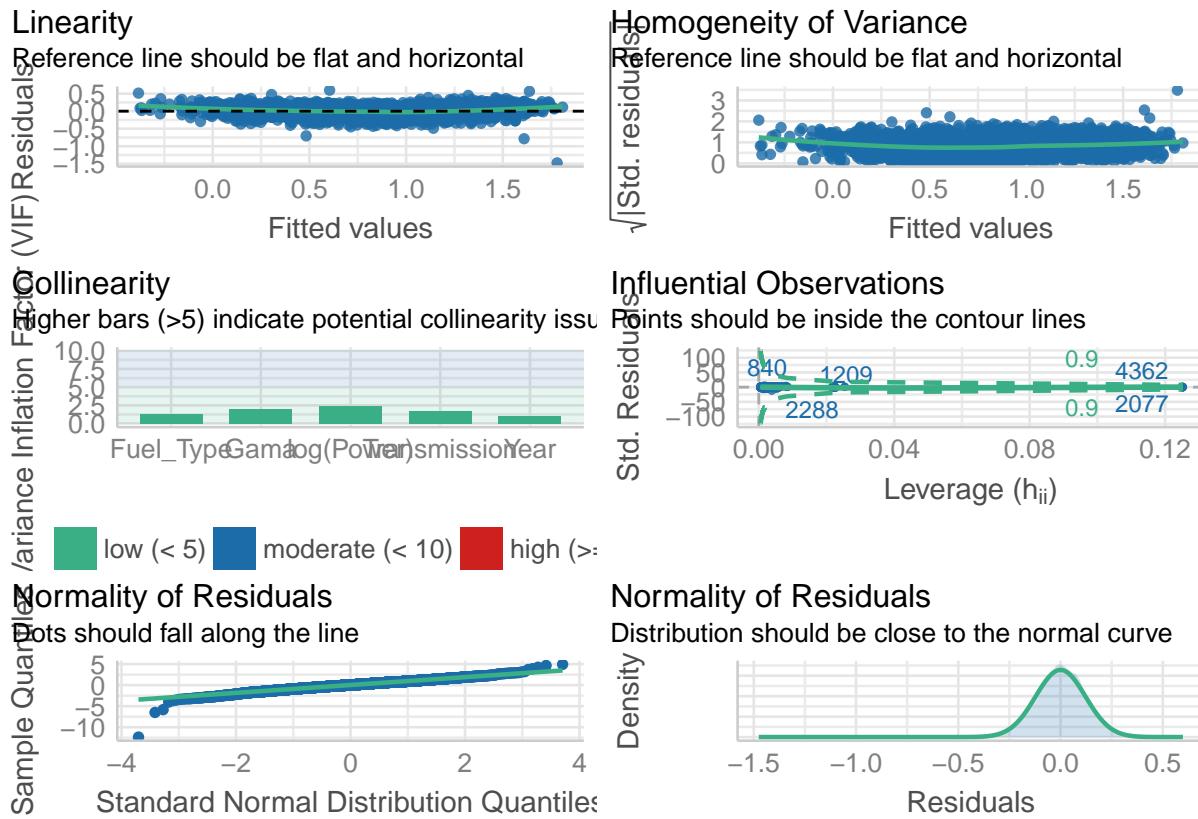
```

##
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + Transmission +
##     log(Power) + Gama, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.47551 -0.07316  0.00337  0.08014  0.59827
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -1.058e+02  1.136e+00 -93.096 <2e-16 ***
## Year                  5.200e-02  5.636e-04  92.267 <2e-16 ***
## Fuel_TypeDiesel       4.532e-02  1.835e-02   2.469  0.0136 *
## Fuel_TypeLPG          4.455e-02  4.665e-02   0.955  0.3397
## Fuel_TypePetrol        -4.382e-02 1.823e-02  -2.404  0.0163 *
## TransmissionManual   -1.116e-01  5.184e-03 -21.532 <2e-16 ***
## log(Power)            4.469e-01  6.609e-03  67.614 <2e-16 ***
## GamaGama baja         -3.004e-01  9.075e-03 -33.100 <2e-16 ***
## GamaGama media        -1.696e-01  5.546e-03 -30.589 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1216 on 4702 degrees of freedom
## Multiple R-squared:  0.8959, Adjusted R-squared:  0.8957
## F-statistic:  5059 on 8 and 4702 DF,  p-value: < 2.2e-16

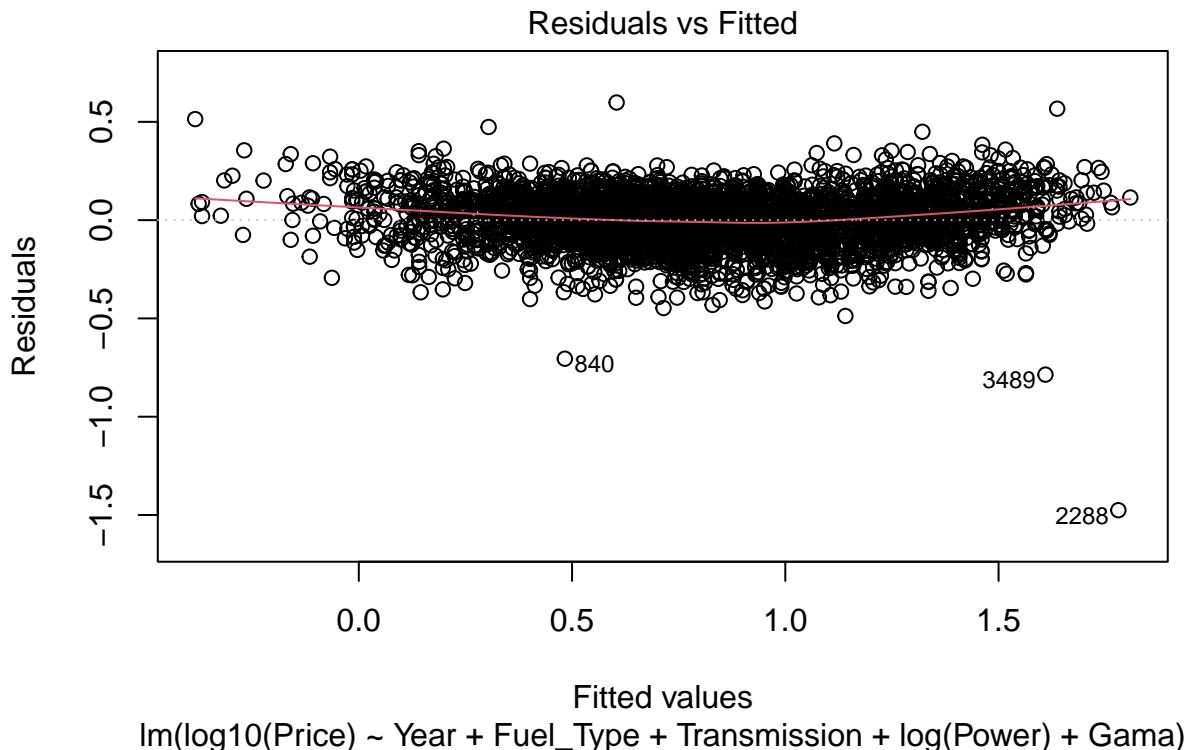
```

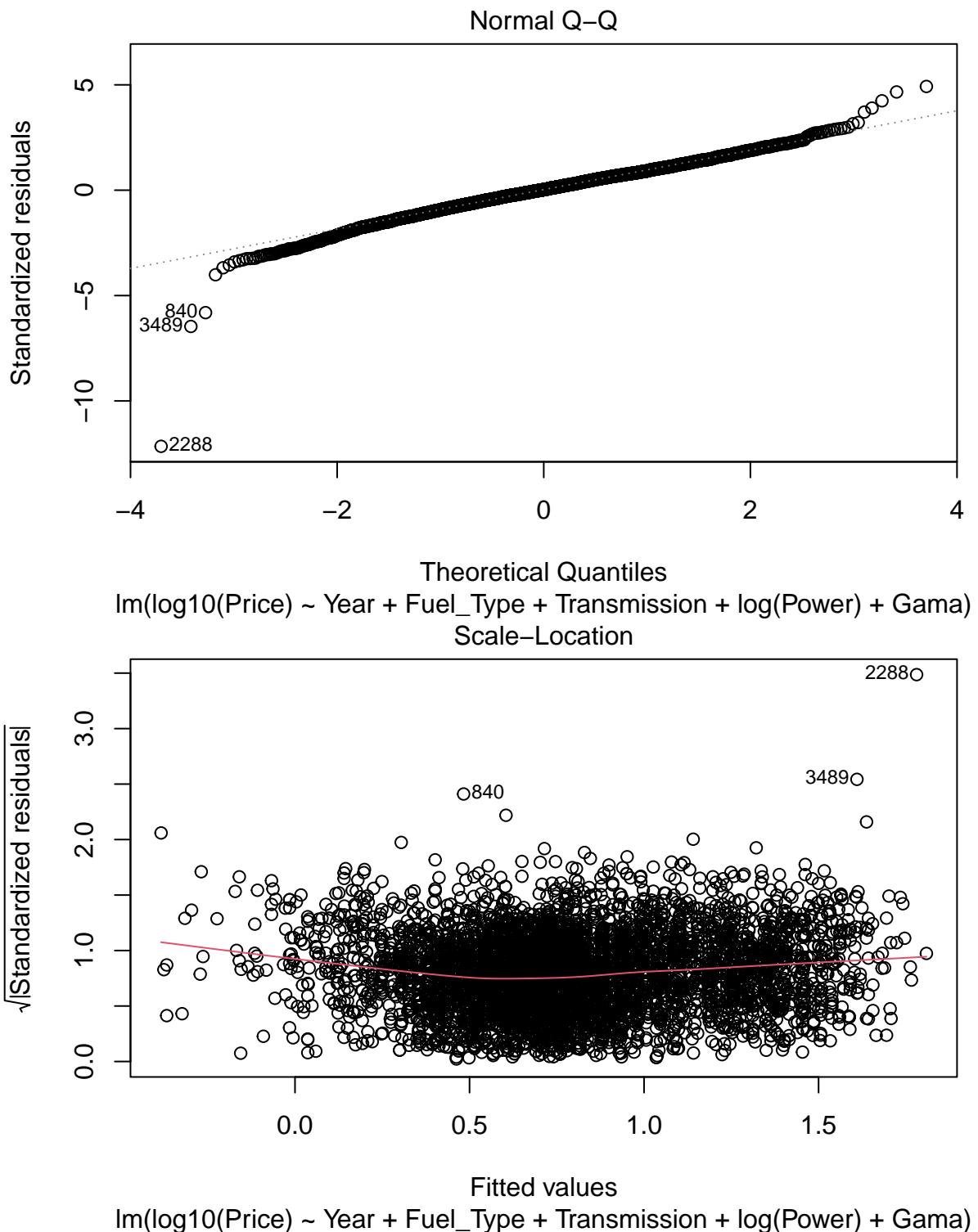
Después de construir el modelo de regresión, con la variable transformada, se observa que el valor del coeficiente relacionado con la variable Power es mayor en 1 unidad. Además, el ajuste mejora considerablemente pasando de un $R^2_{adj}=0.89$ a $R^2_{adj}=0.896$. Haciendo una diagnóstico del modelo:

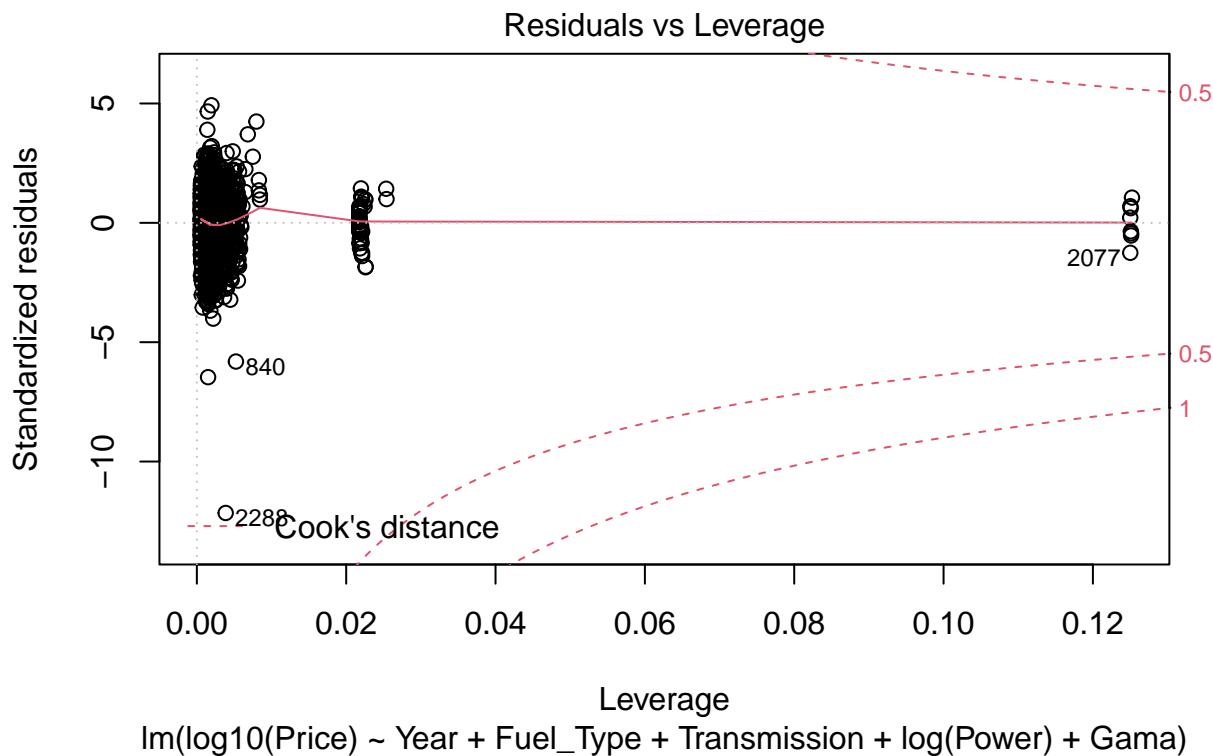
```
check_model(mod7)
```



```
plot(mod7)
```







En el gráfico de *Residuos vs Ajustes* se observa que la media de los residuos es prácticamente cero, luego la linealidad del modelo no se viola.

La *varianza* de los residuos es constante por lo que tampoco se viola la homocedasticidad del modelo, el ajuste es bueno. Lo que si se observa es la aparición de puntos candidatos a ser outliers.

Con el *Q-Q Plot* vemos que los residuos siguen una distribución normal excepto en las colas que varía un poco. Por tanto no se puede asumir que los estimadores de los coeficientes tengan una distribución normal.

En el gráfico de *Residuals vs Leverage* observamos que hay puntos con alto leverage pero ninguno cae fuera de los límites de la distancia de Cook.

```
dwtest(mod7, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data: mod7
## DW = 1.9953, p-value = 0.8722
## alternative hypothesis: true autocorrelation is not 0
vif(mod7)

##          GVIF Df GVIF^(1/(2*Df))
## Year      1.030978  1      1.015371
## Fuel_Type 1.277226  3      1.041625
## Transmission 1.748550  1      1.322328
## log(Power) 2.421325  1      1.556061
## Gama       2.026046  2      1.193060
```

Además, gracias al *Durbin Watson test* sabemos que no hay evidencias de autocorrelación. Utilizamos el VIF y el gráfico de “Collinearity” para analizar la colinealidad de las variables y vemos que no tenemos valores elevados, excepto entre los diferentes tipos de fuente de gasolina entre ellos, por tanto no hay problemas de

multicolinealidad.

Parece que no hay violaciones a los supuestos de OLS. Nuestro modelo parece decente, con un R ajustado de aproximadamente el 90%. Realmente bueno ya para un modelo lineal.

Se va a analizar ahora el siguiente modelo, el **modelo 6**:

```
summary(mod6)
```

```
##  
## Call:  
## lm(formula = log10(Price) ~ Year + Transmission + Power + Seats +  
##      Gama, data = train)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.81452 -0.07933  0.00619  0.08765  0.64034  
##  
## Coefficients:  
##                               Estimate Std. Error t value Pr(>|t|)  
## (Intercept)          -1.103e+02  1.236e+00 -89.240 < 2e-16 ***  
## Year                  5.504e-02  6.137e-04  89.682 < 2e-16 ***  
## TransmissionManual -1.038e-01  5.912e-03 -17.559 < 2e-16 ***  
## Power                 3.521e-03  5.587e-05  63.014 < 2e-16 ***  
## Seats                  1.750e-02  2.703e-03   6.474 1.05e-10 ***  
## GamaGama baja     -3.368e-01  1.010e-02 -33.333 < 2e-16 ***  
## GamaGama media    -1.952e-01  6.489e-03 -30.091 < 2e-16 ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.1335 on 4704 degrees of freedom  
## Multiple R-squared:  0.8746, Adjusted R-squared:  0.8744  
## F-statistic:  5466 on 6 and 4704 DF,  p-value: < 2.2e-16
```

Gracias al análisis de antes, podemos meter Power transformada como logaritmo para permitirle esa linealidad con el precio.

```
mod6=lm(log10(Price) ~ Year+ Transmission+log(Power)+Seats+Gama,data=train)  
summary(mod6)
```

```
##  
## Call:  
## lm(formula = log10(Price) ~ Year + Transmission + log(Power) +  
##      Seats + Gama, data = train)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -1.57049 -0.07939  0.00233  0.08416  0.58651  
##  
## Coefficients:  
##                               Estimate Std. Error t value Pr(>|t|)  
## (Intercept)          -1.089e+02  1.183e+00 -92.048 <2e-16 ***  
## Year                  5.352e-02  5.877e-04  91.065 <2e-16 ***  
## TransmissionManual -9.536e-02  5.667e-03 -16.828 <2e-16 ***  
## log(Power)           4.700e-01  6.821e-03  68.901 <2e-16 ***  
## Seats                 1.008e-03  2.590e-03   0.389   0.697  
## GamaGama baja     -3.185e-01  9.720e-03 -32.771 <2e-16 ***
```

```

## GamaGama media      -2.024e-01  6.101e-03 -33.182   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1279 on 4704 degrees of freedom
## Multiple R-squared:  0.8849, Adjusted R-squared:  0.8847
## F-statistic:  6026 on 6 and 4704 DF,  p-value: < 2.2e-16

```

Al transformar Power para lograr linealidad, se aprecia que la variable Seats deja de ser significativa por lo que procedemos a eliminarla:

```

mod6=lm(log10(Price) ~ Year+ Transmission+log(Power)+Gama,data=train)
summary(mod6)

```

```

##
## Call:
## lm(formula = log10(Price) ~ Year + Transmission + log(Power) +
##     Gama, data = train)
##
## Residuals:
##       Min     1Q Median     3Q    Max
## -1.57116 -0.07953  0.00228  0.08419  0.58639
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.089e+02  1.183e+00 -92.09   <2e-16 ***
## Year         5.353e-02  5.873e-04   91.14   <2e-16 ***
## TransmissionManual -9.468e-02  5.390e-03  -17.57   <2e-16 ***
## log(Power)    4.701e-01  6.812e-03   69.01   <2e-16 ***
## GamaGama baja -3.193e-01  9.494e-03  -33.63   <2e-16 ***
## GamaGama media -2.034e-01  5.611e-03  -36.25   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1279 on 4705 degrees of freedom
## Multiple R-squared:  0.8849, Adjusted R-squared:  0.8847
## F-statistic:  7232 on 5 and 4705 DF,  p-value: < 2.2e-16

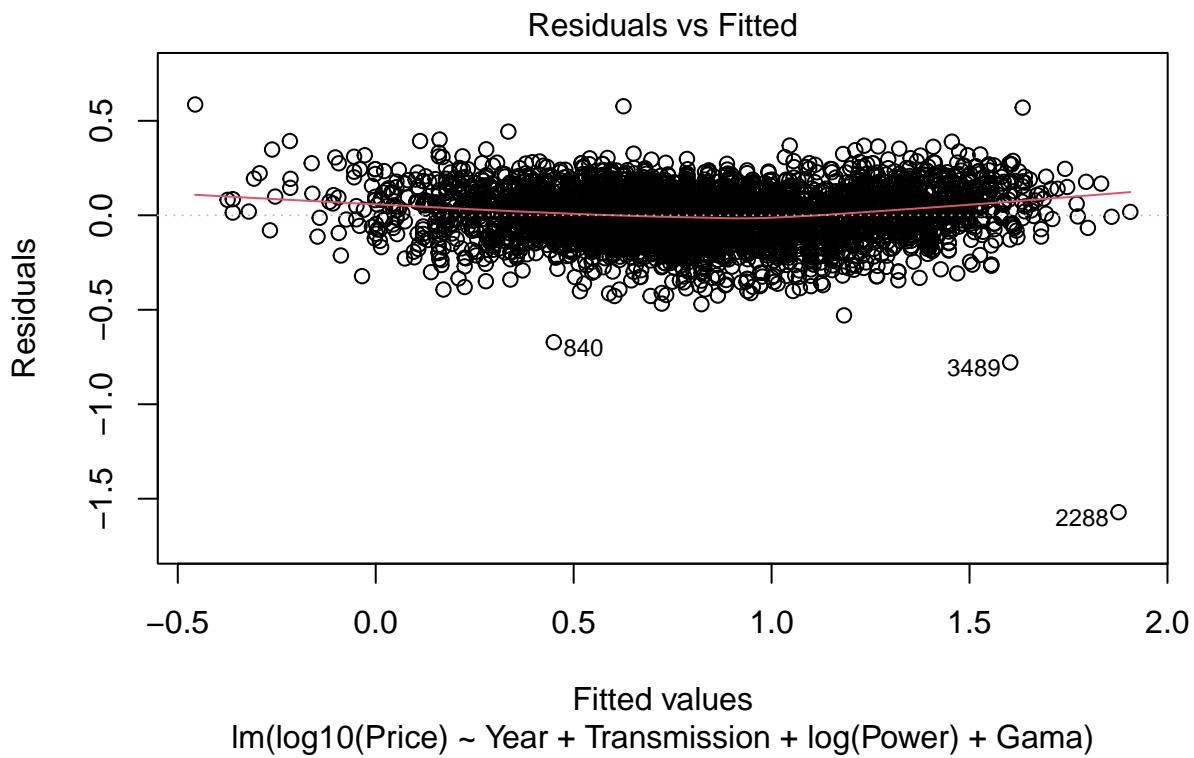
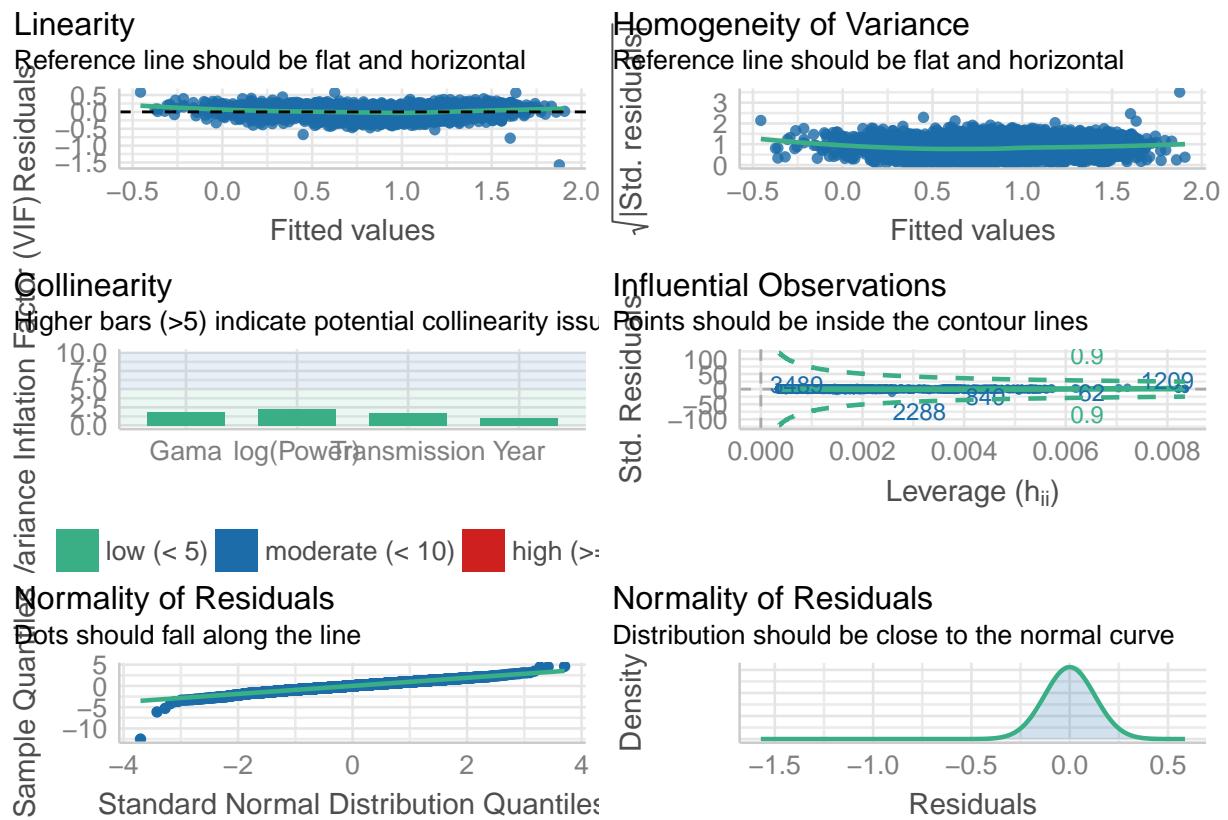
```

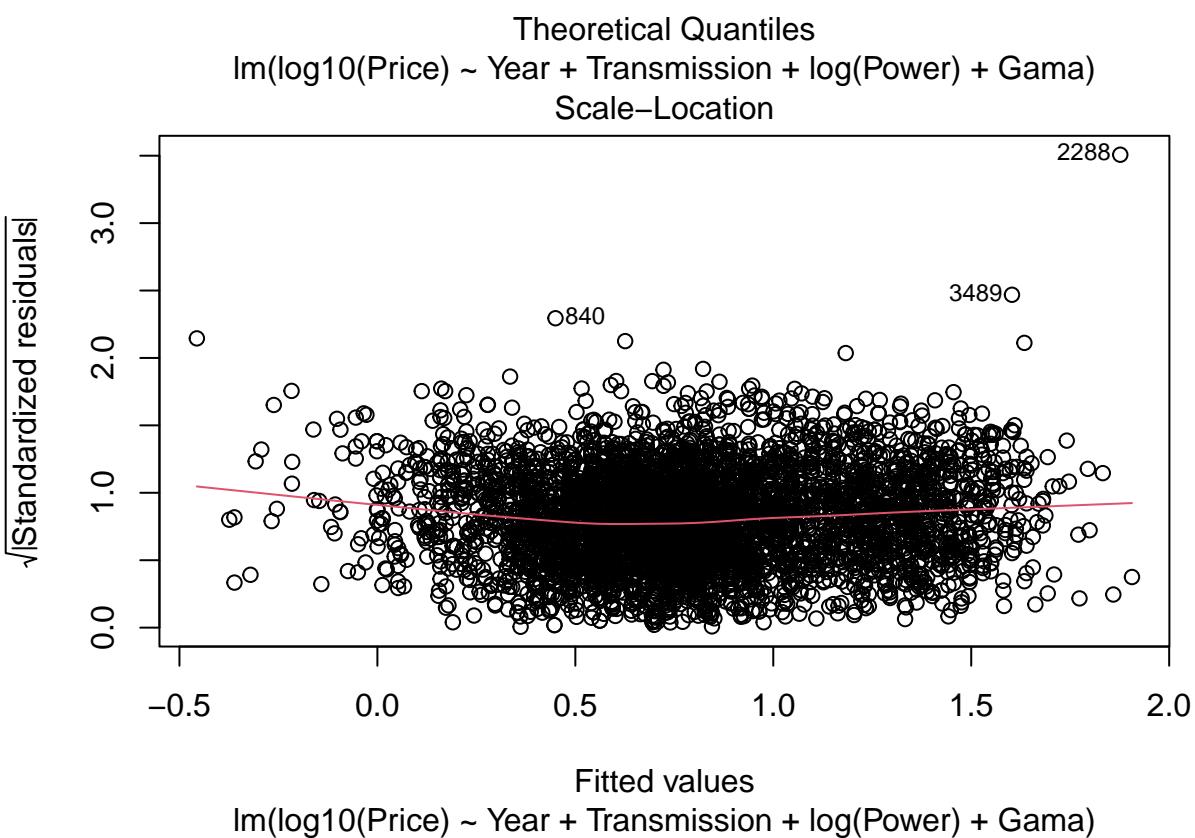
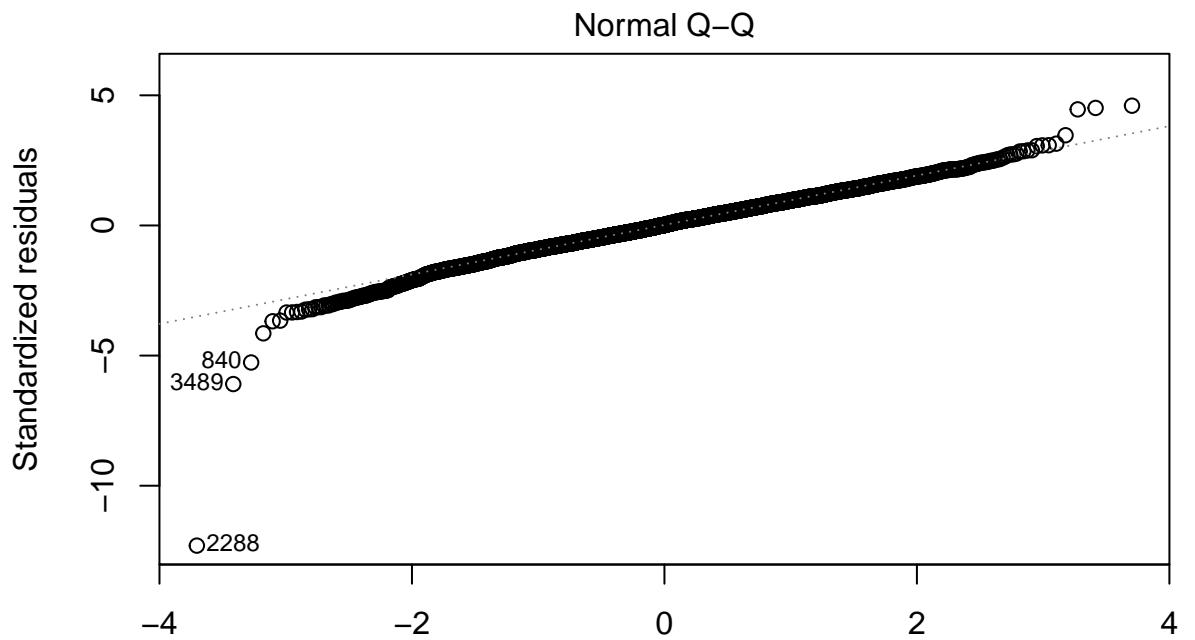
Después de construir el modelo de regresión, con la variable transformada, se observa que el valor del coeficiente relacionado con la variable Power es mayor en 1 unidad. Además, el ajuste mejora considerablemente pasando de un $R^2_{adj}=0.874$ a $R^2_{adj}=0.885$. Haciendo una diagnóstico del modelo:

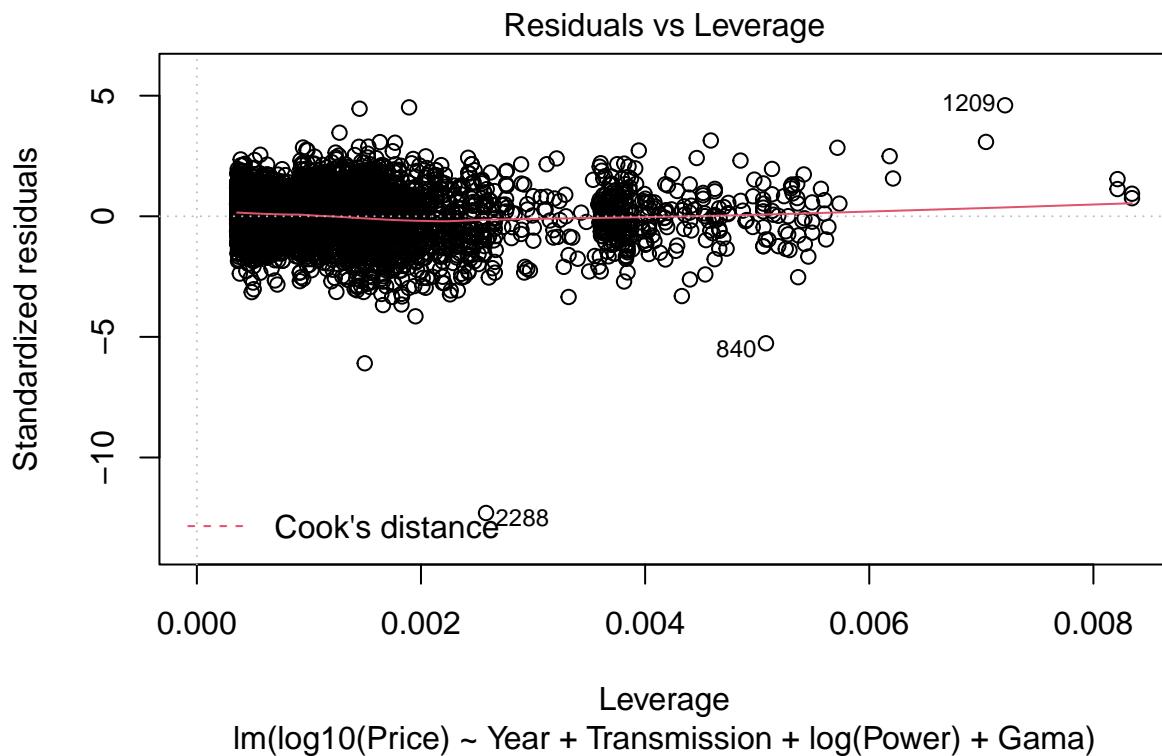
```

check_model(mod6)

```







En el gráfico de *Residuos vs Ajustes* se observa que la media de los residuos es prácticamente cero, luego la linealidad del modelo no se viola.

La *varianza* de los residuos es constante por lo que tampoco se viola la homocedasticidad del modelo, el ajuste es bueno. Lo que si se observa es la aparición de puntos candidatos a ser outliers.

Con el *Q-Q Plot* vemos que los residuos siguen una distribución normal excepto en las colas que varía un poco. Por tanto no se puede asumir que los estimadores de los coeficientes tengan una distribución normal.

En el gráfico de *Residuals vs Leverage* observamos que hay puntos con alto leverage pero ninguno cae fuera de los límites de la distancia de Cook.

```
dwtest(mod6, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data: mod6
## DW = 1.9787, p-value = 0.4663
## alternative hypothesis: true autocorrelation is not 0
vif(mod6)

##          GVIF Df GVIF^(1/(2*Df))
## Year      1.012883  1      1.006421
## Transmission 1.709927  1      1.307642
## log(Power)  2.326752  1      1.525369
## Gama       1.864808  2      1.168580
```

Además, gracias al *Durbin Watson test* sabemos que no hay evidencias de autocorrelación. Utilizamos el VIF y el gráfico de la colinealidad para comprobar ello y vemos que no tenemos valores elevados, excepto entre los diferentes tipos de fuente de gasolina entre ellos, por tanto no hay problemas de multicolinealidad.

Parece que no hay violaciones a los supuestos de OLS. Nuestro modelo parece decente, con un R cuadrado de

aproximadamente el 90%. Realmente bueno ya para un modelo lineal.

Se va a analizar ahora el siguiente modelo, el **modelo 3**:

```
summary(mod3)

##
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + Power + Gama,
##      data = train)
##
## Residuals:
##    Min      1Q   Median      3Q     Max 
## -1.76614 -0.07477  0.00786  0.08045  0.73831
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.095e+02  1.207e+00 -90.727 < 2e-16 ***
## Year         5.460e-02  5.993e-04  91.106 < 2e-16 ***
## Fuel_TypeDiesel 1.150e-01  1.953e-02  5.888 4.18e-09 ***
## Fuel_TypeLPG   5.006e-02  4.990e-02  1.003   0.316  
## Fuel_TypePetrol 1.455e-02  1.945e-02  0.748   0.455  
## Power        3.892e-03  4.782e-05  81.390 < 2e-16 ***
## GamaGama baja -3.430e-01  9.562e-03 -35.867 < 2e-16 ***
## GamaGama media -1.829e-01  5.931e-03 -30.842 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1301 on 4703 degrees of freedom
## Multiple R-squared:  0.8809, Adjusted R-squared:  0.8807 
## F-statistic:  4968 on 7 and 4703 DF,  p-value: < 2.2e-16
```

Si observamos los p-valores de los test de hipótesis para los coeficientes individuales, observamos que para alguna variable no se puede rechazar que el coeficiente sea cero. Además, gracias al análisis de antes, podemos meter Power transformada como logaritmo para permitirle esa linealidad con el precio.

```
mod3=lm(log10(Price) ~ Year+Fuel_Type+log(Power)+Gama,data=train)
summary(mod3)
```

```
##
## Call:
## lm(formula = log10(Price) ~ Year + Fuel_Type + log(Power) + Gama,
##      data = train)
##
## Residuals:
##    Min      1Q   Median      3Q     Max 
## -1.50378 -0.07255  0.00383  0.08013  0.67509
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.086e+02  1.183e+00 -91.829 <2e-16 ***
## Year         5.322e-02  5.876e-04  90.572 <2e-16 ***
## Fuel_TypeDiesel 3.270e-02  1.923e-02  1.701   0.0890 .  
## Fuel_TypeLPG   5.204e-02  4.889e-02  1.065   0.2872  
## Fuel_TypePetrol -4.372e-02  1.911e-02 -2.288   0.0222 *  
## log(Power)     5.142e-01  6.103e-03  84.250 <2e-16 ***
```

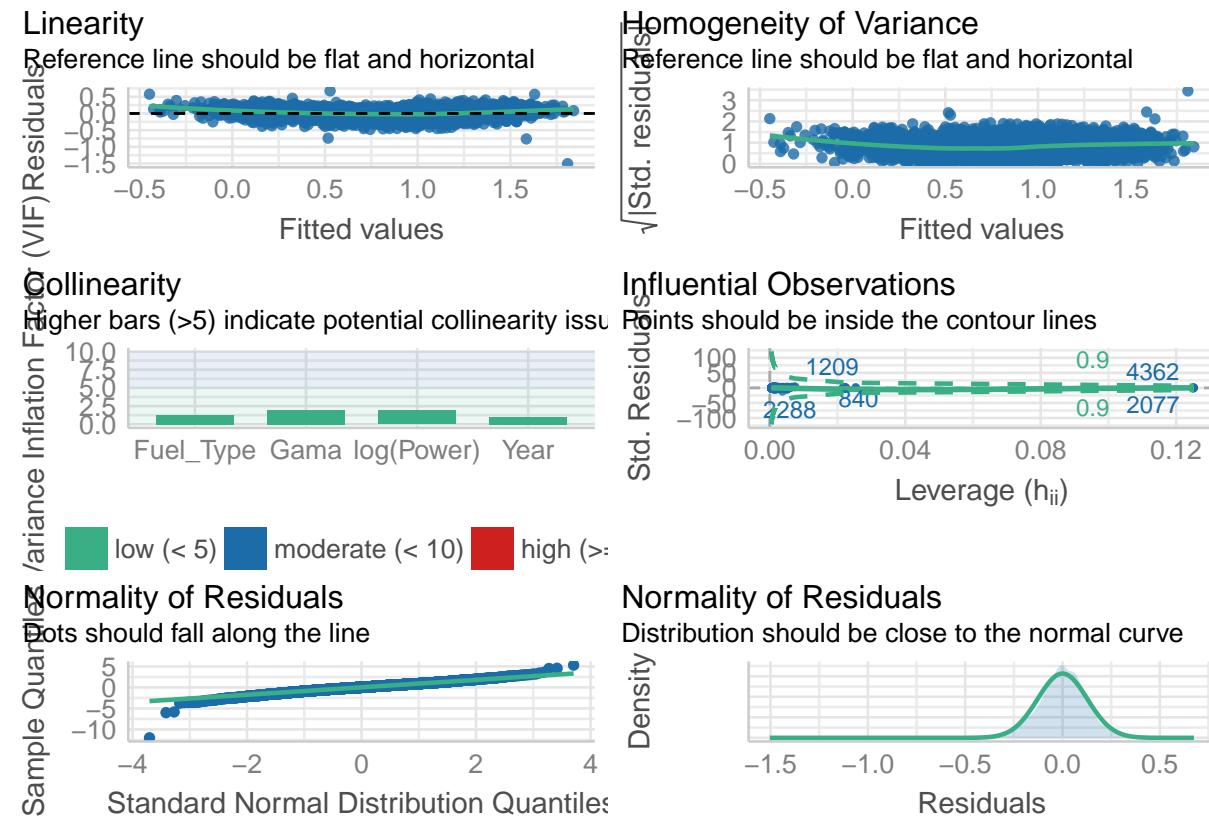
```

## GamaGama baja -3.195e-01 9.465e-03 -33.754 <2e-16 ***
## GamaGama media -1.892e-01 5.734e-03 -32.992 <2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1275 on 4703 degrees of freedom
## Multiple R-squared: 0.8857, Adjusted R-squared: 0.8855
## F-statistic: 5204 on 7 and 4703 DF, p-value: < 2.2e-16

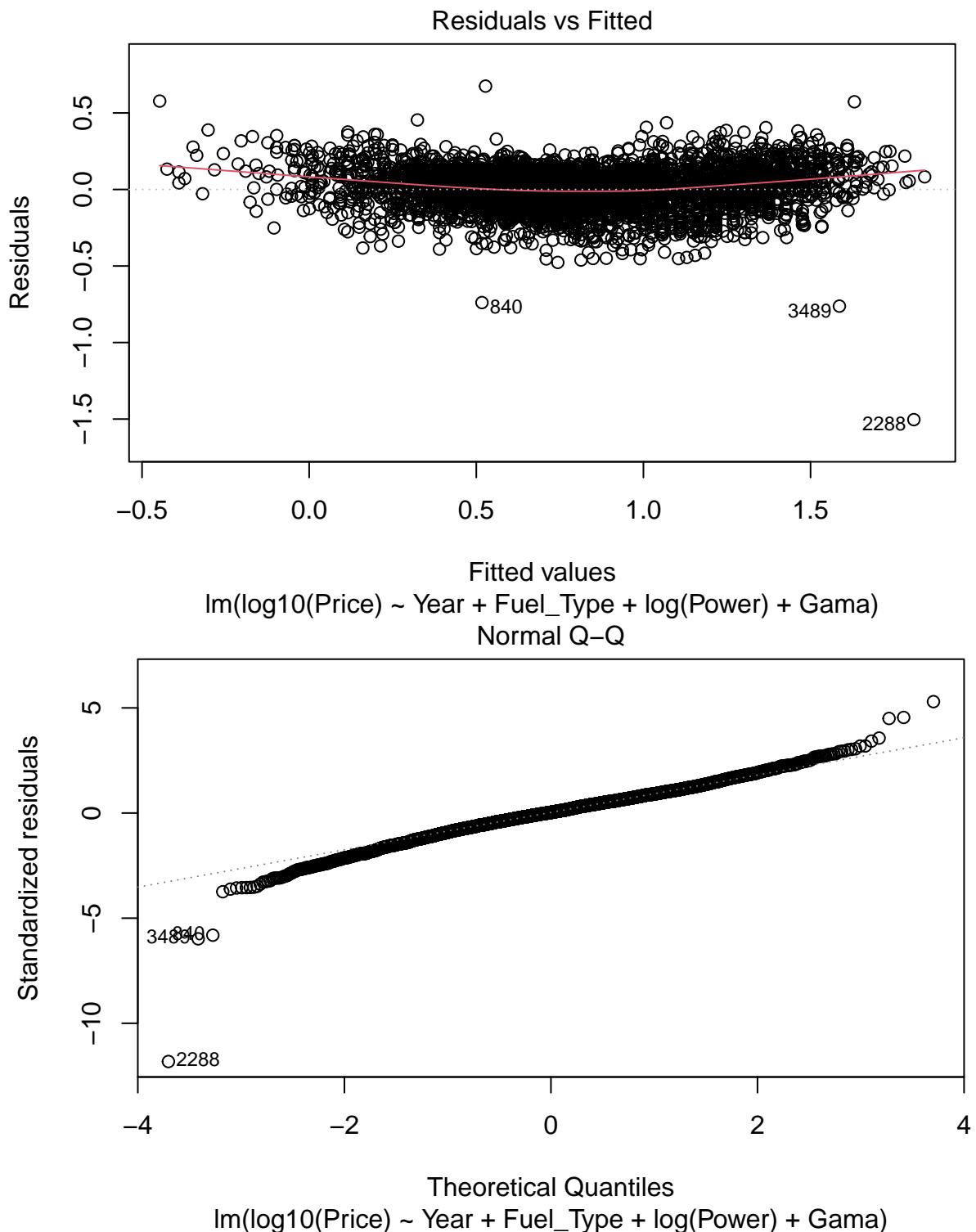
```

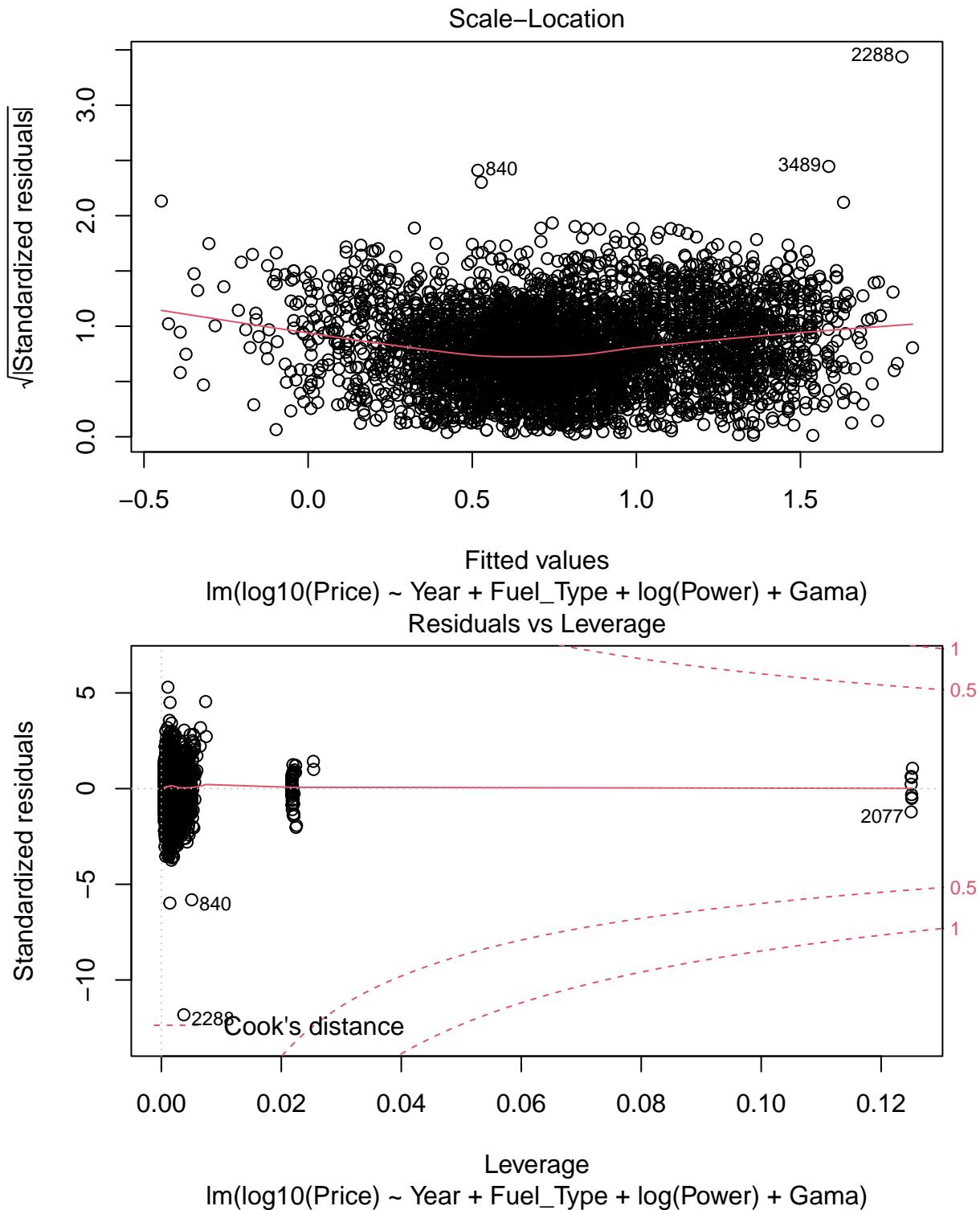
Después de construir el modelo de regresión, con la variable transformada, se observa que el valor del coeficiente relacionado con la variable Power es mayor en 1 unidad. Haciendo una diagnóstico del modelo:

```
check_model(mod3)
```



```
plot(mod3)
```





En el gráfico de *Residuos vs Ajustes* se observa que la media de los residuos es prácticamente cero, luego la linealidad del modelo no se viola.

La varianza de los residuos es constante por lo que tampoco se viola la homocedasticidad del modelo, el ajuste es bueno. Lo que si se observa es la aparición de puntos candidatos a ser outliers.

Con el *Q-Q Plot* vemos que los residuos siguen una distribución normal excepto en las colas que varía un

poco. Por tanto no se puede asumir que los estimadores de los coeficientes tengan una distribución normal. En el gráfico de *Residuals vs Leverage* observamos que hay puntos con alto leverage pero ninguno cae fuera de los límites de la distancia de Cook

```
dwtest(mod3, alternative = "two.sided")

##
## Durbin-Watson test
##
## data: mod3
## DW = 1.9871, p-value = 0.6584
## alternative hypothesis: true autocorrelation is not 0

vif(mod3)

##          GVIF Df GVIF^(1/(2*Df))
## Year      1.020493  1      1.010195
## Fuel_Type 1.249014  3      1.037754
## log(Power) 1.879720  1      1.371029
## Gama      1.971567  2      1.184958
```

Además, gracias al *Durbin Watson test* sabemos que no hay evidencias de autocorrelación. Utilizamos el VIF para comprobar la multicolinealidad y vemos que no tenemos valores elevados, por tanto no hay problemas de multicolinealidad.

Parece que no hay violaciones a los supuestos de OLS. Nuestro modelo parece decente, con un R cuadrado de aproximadamente el 90%. Realmente bueno ya para un modelo lineal.

Se va a analizar ahora el siguiente modelo, el **modelo 2**:

```
summary(mod2)

##
## Call:
## lm(formula = log10(Price) ~ Year + Power + Gama, data = train)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -1.86964 -0.07973  0.00672  0.08632  0.70449
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.126e+02  1.268e+00 -88.83 <2e-16 ***
## Year        5.621e-02  6.299e-04  89.23 <2e-16 ***
## Power       3.947e-03  5.048e-05  78.19 <2e-16 ***
## GamaGama baja -3.679e-01  1.006e-02 -36.56 <2e-16 ***
## GamaGama media -2.243e-01  6.009e-03 -37.32 <2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1378 on 4706 degrees of freedom
## Multiple R-squared:  0.8662, Adjusted R-squared:  0.8661 
## F-statistic: 7620 on 4 and 4706 DF,  p-value: < 2.2e-16
```

Gracias al análisis de antes, podemos meter Power transformada como logaritmo para permitirle esa linealidad con el precio.

```

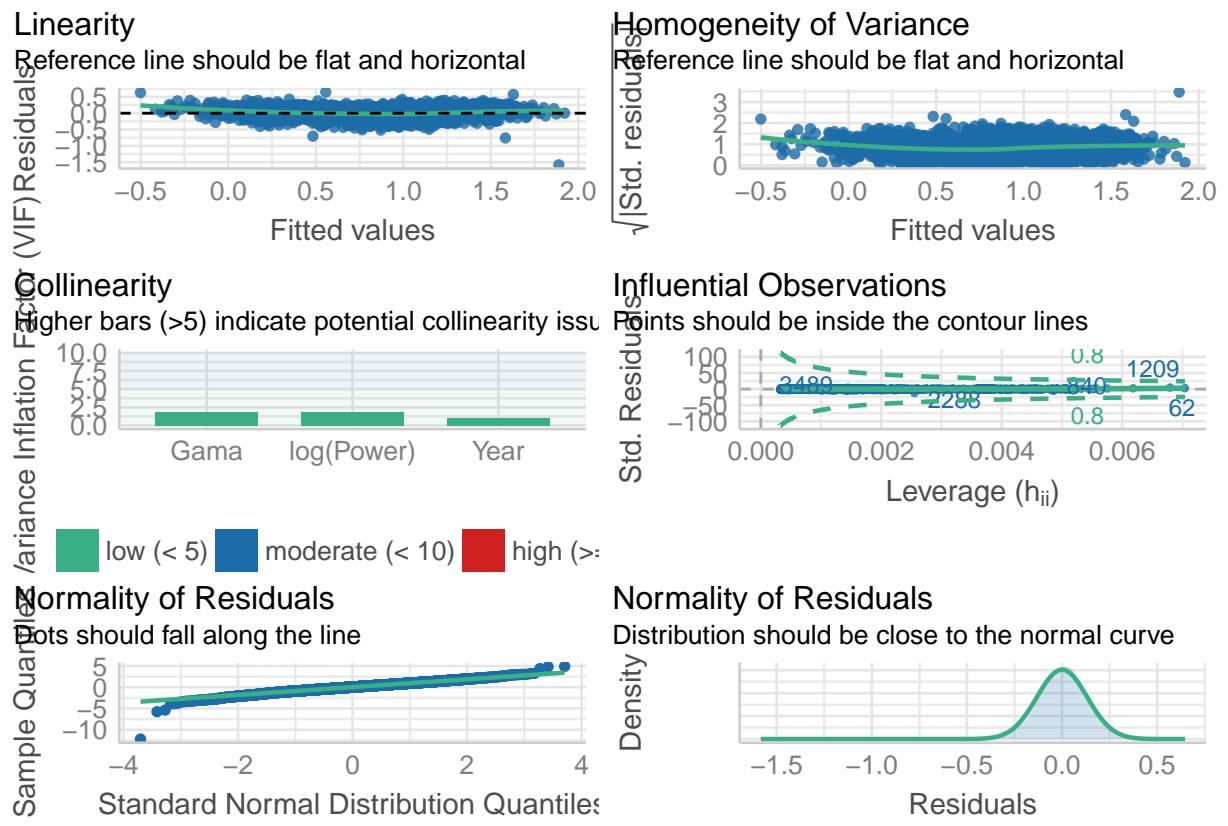
mod2=lm(log10(Price) ~ Year+log(Power)+Gama,data=train)
summary(mod2)

##
## Call:
## lm(formula = log10(Price) ~ Year + log(Power) + Gama, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.58359 -0.07878  0.00429  0.08516  0.64611 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.110e+02  1.215e+00 -91.36   <2e-16 ***
## Year        5.440e-02  6.041e-04   90.05   <2e-16 ***
## log(Power)   5.253e-01  6.239e-03   84.20   <2e-16 ***
## GamaGama baja -3.337e-01  9.763e-03  -34.18   <2e-16 ***
## GamaGama media -2.162e-01  5.742e-03  -37.65   <2e-16 ***  
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.132 on 4706 degrees of freedom
## Multiple R-squared:  0.8773, Adjusted R-squared:  0.8772 
## F-statistic:  8413 on 4 and 4706 DF,  p-value: < 2.2e-16

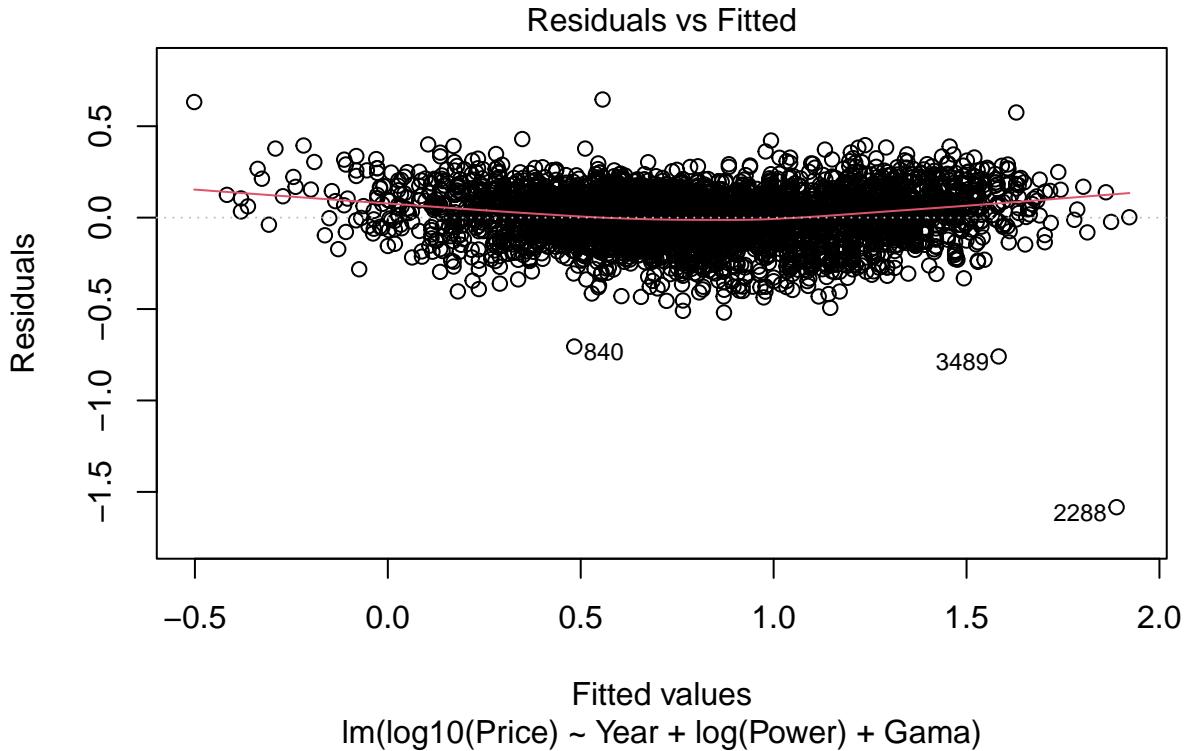
```

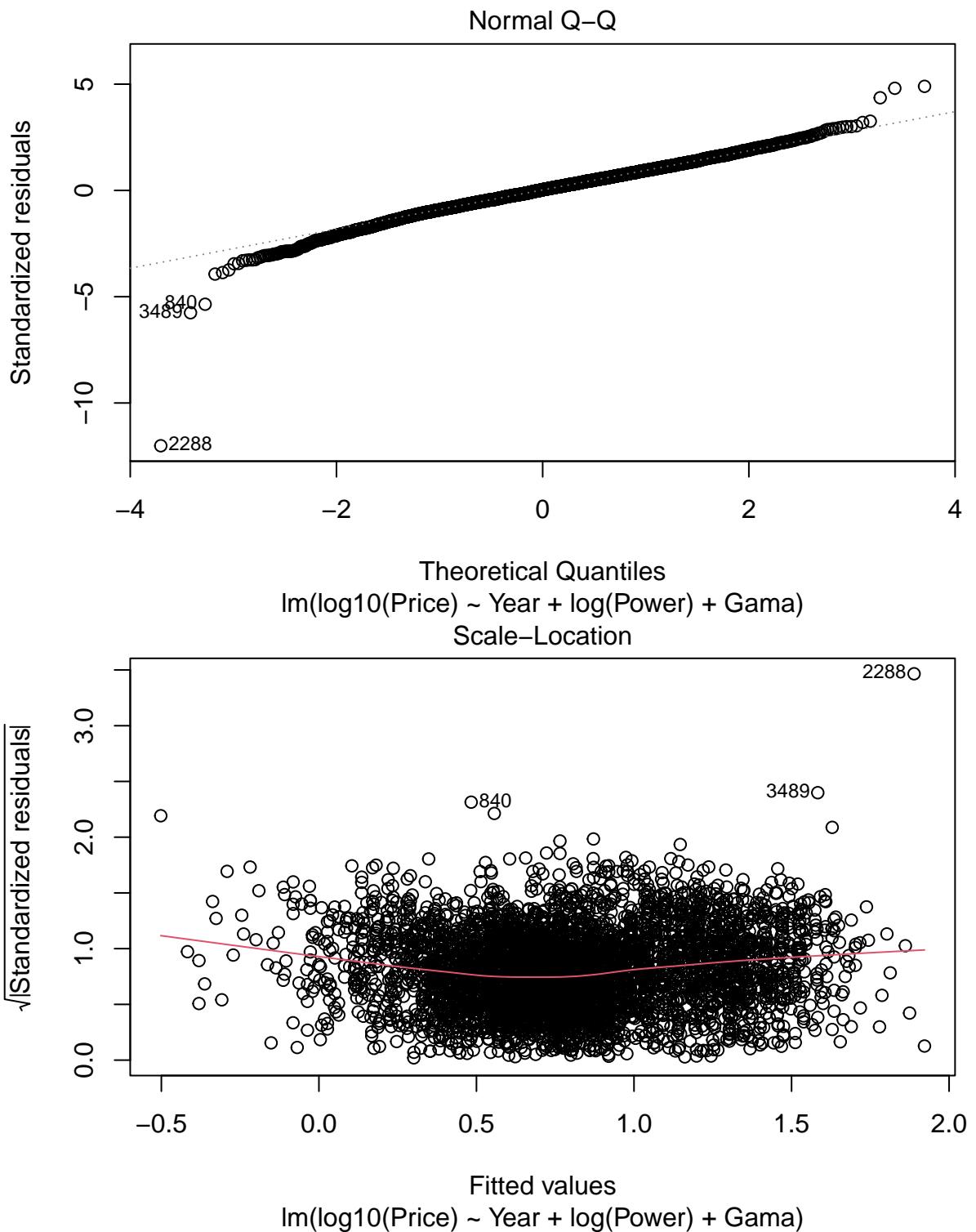
Después de construir el modelo de regresión, con la variable transformada, se observa que el valor del coeficiente relacionado con la variable Power es mayor en 1 unidad. Además, el ajuste mejora considerablemente pasando de un $R^2_{adj}=0.866$ a $R^2_{adj}=0.877$. Haciendo una diagnóstico del modelo:

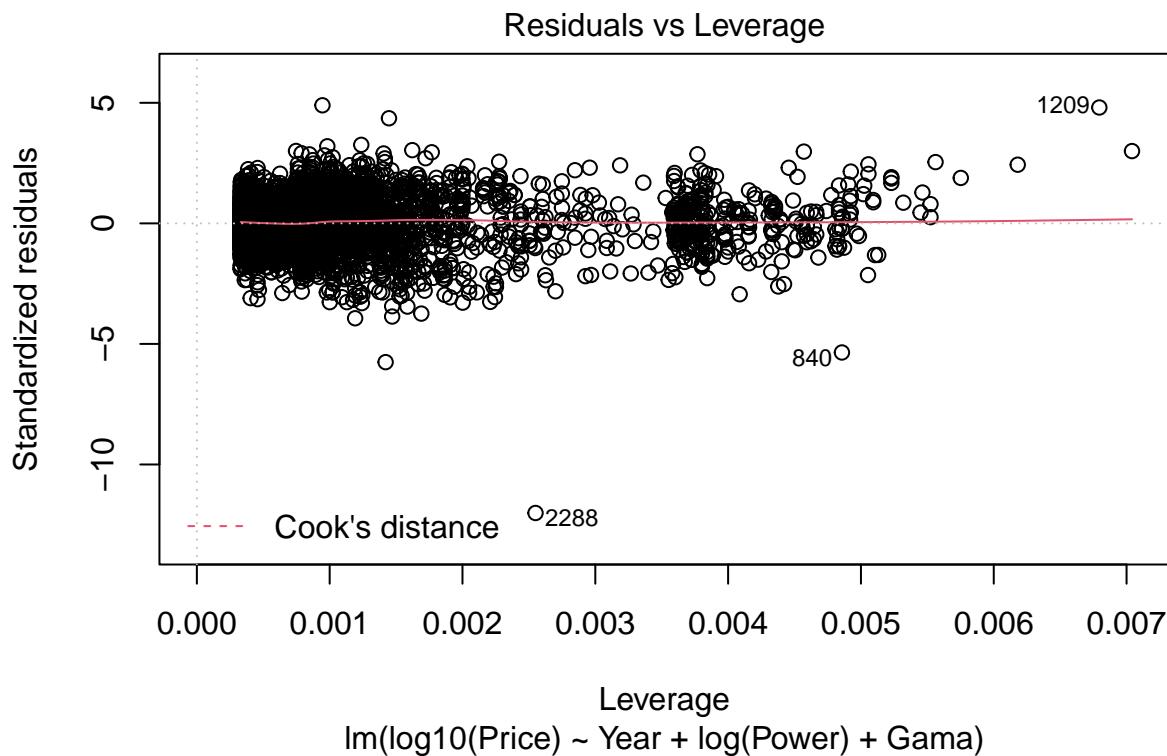
```
check_model(mod2)
```



```
plot(mod2)
```







En el gráfico de *Residuos vs Ajustes* se observa que la media de los residuos es prácticamente cero, luego la linealidad del modelo no se viola.

La *varianza* de los residuos es constante por lo que tampoco se viola la homocedasticidad del modelo, el ajuste es bueno. Lo que si se observa es la aparición de puntos candidatos a ser outliers.

Con el *Q-Q Plot* vemos que los residuos siguen una distribución normal excepto en las colas que varía un poco. Por tanto no se puede asumir que los estimadores de los coeficientes tengan una distribución normal.

En el gráfico de *Residuals vs Leverage* observamos que hay puntos con alto leverage pero ninguno cae fuera de los límites de la distancia de Cook.

```
dwtest(mod2, alternative = "two.sided")
```

```
##  
## Durbin-Watson test  
##  
## data: mod2  
## DW = 1.9755, p-value = 0.4008  
## alternative hypothesis: true autocorrelation is not 0  
vif(mod2)
```

```
##          GVIF Df GVIF^(1/(2*Df))  
## Year      1.005718  1      1.002855  
## log(Power) 1.831851  1      1.353459  
## Gama      1.832525  2      1.163490
```

```
ols_coll_diag(mod2)
```

```
## Tolerance and Variance Inflation Factor  
## -----  
##          Variables Tolerance      VIF  
## 1           Year 0.9943147 1.005718
```

```

## 2      log(Power) 0.5458960 1.831851
## 3  GamaGama baja 0.6965017 1.435747
## 4 GamaGama media 0.4955476 2.017970
##
##
## Eigenvalue and Condition Index
## -----
##   Eigenvalue Condition Index    intercept          Year  log(Power)
## 1 3.750474e+00     1.000000 1.758253e-07 1.754455e-07 3.021427e-04
## 2 1.000020e+00     1.936595 8.307491e-12 7.567443e-12 6.066499e-08
## 3 2.467753e-01     3.898452 5.143531e-07 5.133715e-07 2.185087e-03
## 4 2.729691e-03     37.066902 1.714036e-04 1.700308e-04 9.957302e-01
## 5 1.251748e-06    1730.950379 9.998279e-01 9.998293e-01 1.782460e-03
##   GamaGama baja GamaGama media
## 1  0.0030838940   0.0090595575
## 2  0.6016246513   0.0140430883
## 3  0.1678885001   0.4957427421
## 4  0.2268522607   0.4803236003
## 5  0.0005506939   0.0008310117

```

Además, gracias al *Durbin Watson test* sabemos que no hay evidencias de autocorrelación. Utilizamos el VIF para comprobar la colinealidad y vemos que no tenemos valores elevados, por tanto no hay problemas de multicolinealidad.

Parece que no hay violaciones a los supuestos de OLS. Nuestro modelo parece decente, con un R cuadrado de aproximadamente el 90%. Realmente bueno ya para un modelo lineal.

De nuestro análisis hemos obtenido información:

- El año de fabricación del coche y la gama en función de la marca dan una indicación visual y un efecto significativo en el precio.

- La potencia se correlaciona mejor con el precio que con el motor.

- Además de estas variables, no se muestra ninguna indicación clara en las demás.

Se decide que **el mejor modelo para explicar nuestros datos es este, el modelo 2**. Vamos a analizar algunos parámetros más que serán importantes y, aunque se trata de un modelo bueno, tiene defectos que se solucionarían haciendo que el modelo deje de ser lineal.

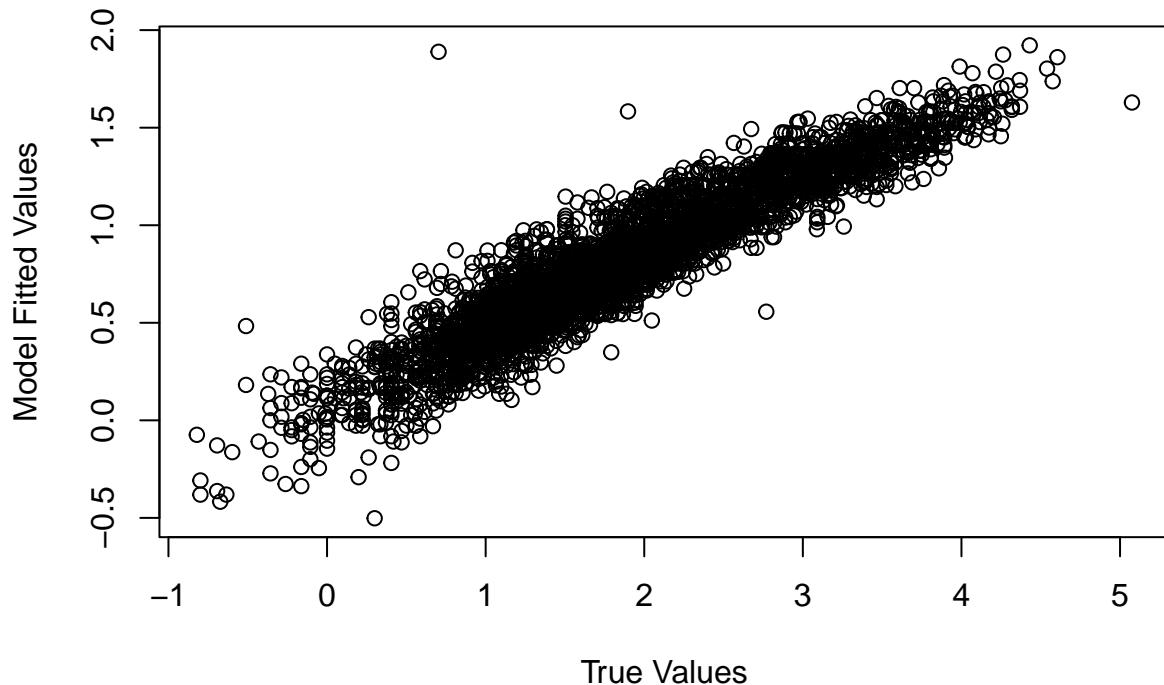
Ahora, trataremos los valores ajustados y los reales de nuestros datos. Los valores ajustados (o predichos) son los valores de y que esperaría para los valores de x dados de acuerdo con el modelo de regresión construido (o visualmente, la línea de regresión recta de mejor ajuste).

```

train$Price.mod2 <- mod2$fitted.values
plot(x = log(train$Price),                      # True values on x-axis
      y = mod2$fitted.values,                   # fitted values on y-axis
      xlab = "True Values",
      ylab = "Model Fitted Values",
      main = "Regression fits")

```

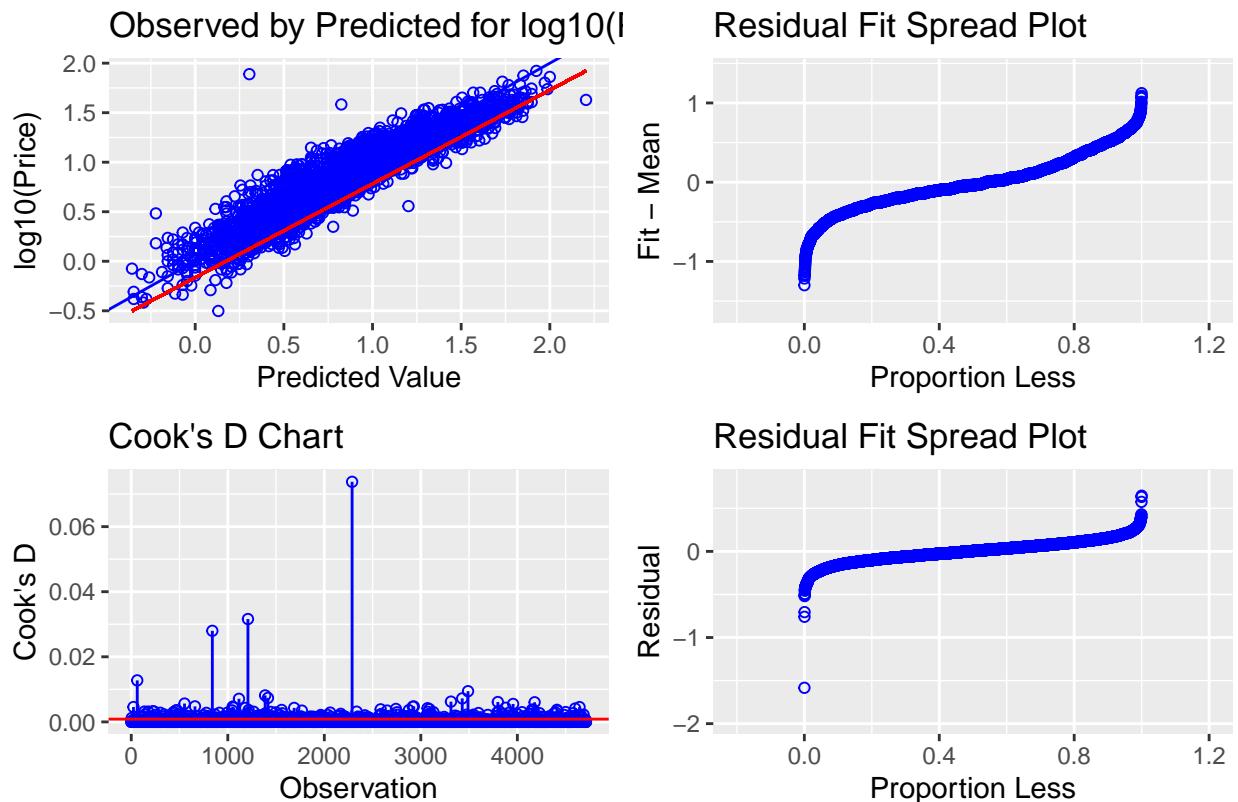
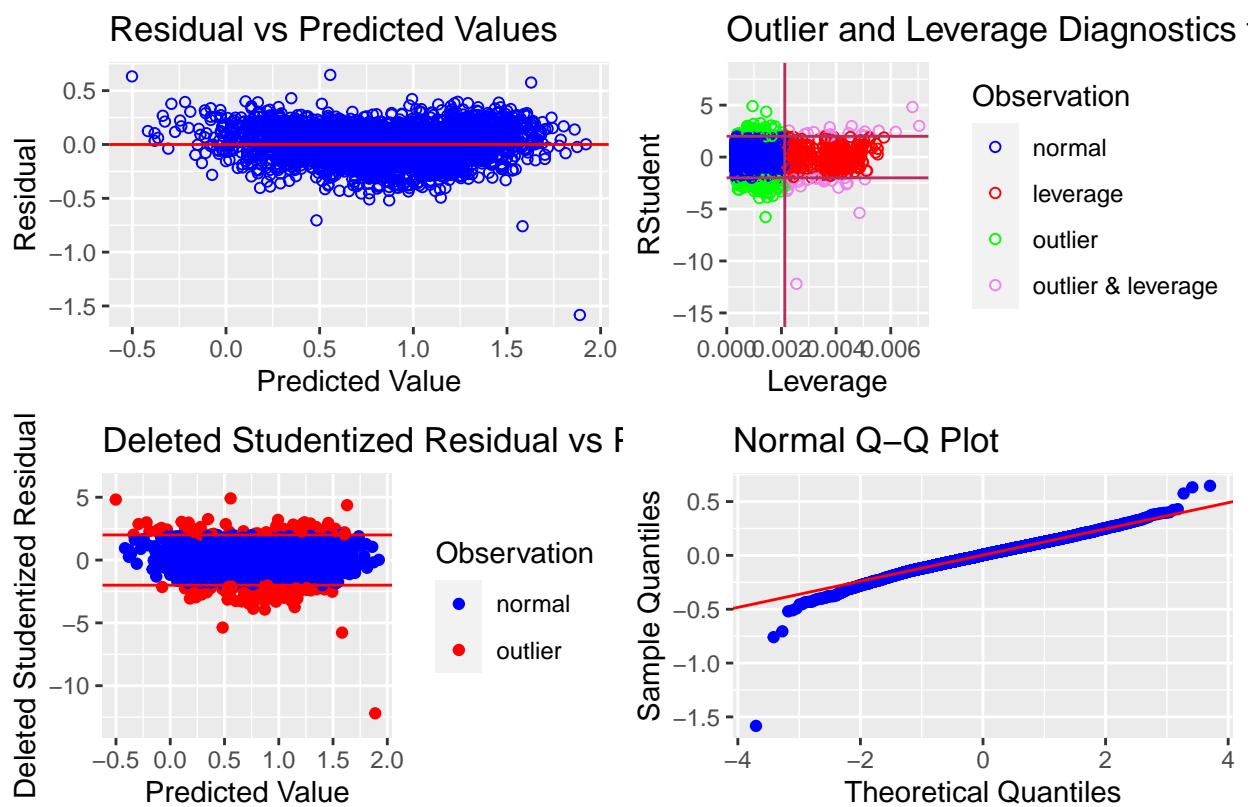
Regression fits



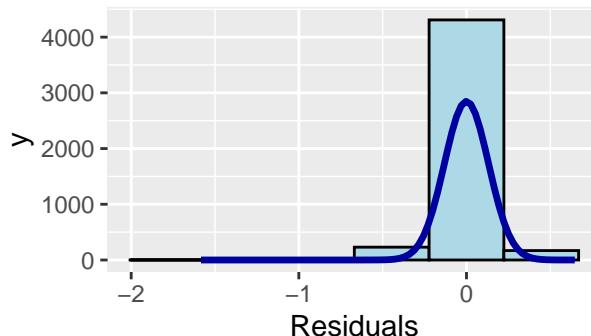
En el diagrama de dispersión ilustrado, se puede ver que no todos los puntos de datos caen exactamente en la línea de regresión estimada. Esto significa que, para un determinado coche, los valores del precio pueden ser diferentes de los valores de precio previstos. La diferencia se llama errores residuales. Se ve sobretodo una dispersión en los coches de precio más bajo.

A continuación vamos a estudiar los residuos del modelo. El estudio de residuos es una herramienta formidable en el estudio de las regresiones lineales. Nos sirve para saber si se están cumpliendo las premisas de linealidad de las relaciones, homocedasticidad y normalidad de los residuos. Para una mejor visualización de los residuos del modelo, veamos los siguientes gráficos:

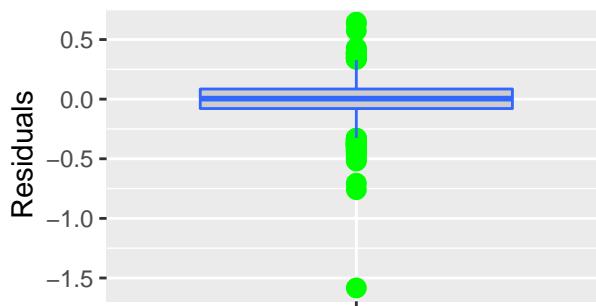
```
ols_plot_diagnostics(mod2)
```



Residual Histogram



Residual Box Plot



En cuanto a la normalidad de los residuos, en el histograma se ve un aspecto general normal. Por supuesto no reproduce exactamente la curva normal, pero podemos atribuir esas desviaciones. Además, como dijimos anteriormente, en el Q-Q Plot se ajusta bien a la recta excepto en los extremos.

```
residuos=resid(mod2)
ks.test(residuos, 'pnorm')
```

```
## Warning in ks.test(residuos, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test

##
## One-sample Kolmogorov-Smirnov test
##
## data: residuos
## D = 0.37845, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Se contrasta la hipótesis nula de que los residuos del modelo se distribuyen según una distribución Normal. Al obtener un p.valor menor que 0.05 rechazamos H₀ y por consiguiente se tiene que los residuos no siguen una distribución Normal. Sin embargo, debido a que los modelos lineales en general son robustos, proseguiremos con el análisis asumiendo cierta normalidad. Obtenemos:

-Valores ajustados: valores ajustados (valores de la variable respuesta) para las observaciones originales de la predictoría.

-Residuos: diferencia entre valor observado de la respuesta y valor ajustado por el modelo.

-Estadísticos: residuos estandarizados del modelo ajustado.

Ahora, vamos a analizar si existen observaciones influyentes y/o anómalas:

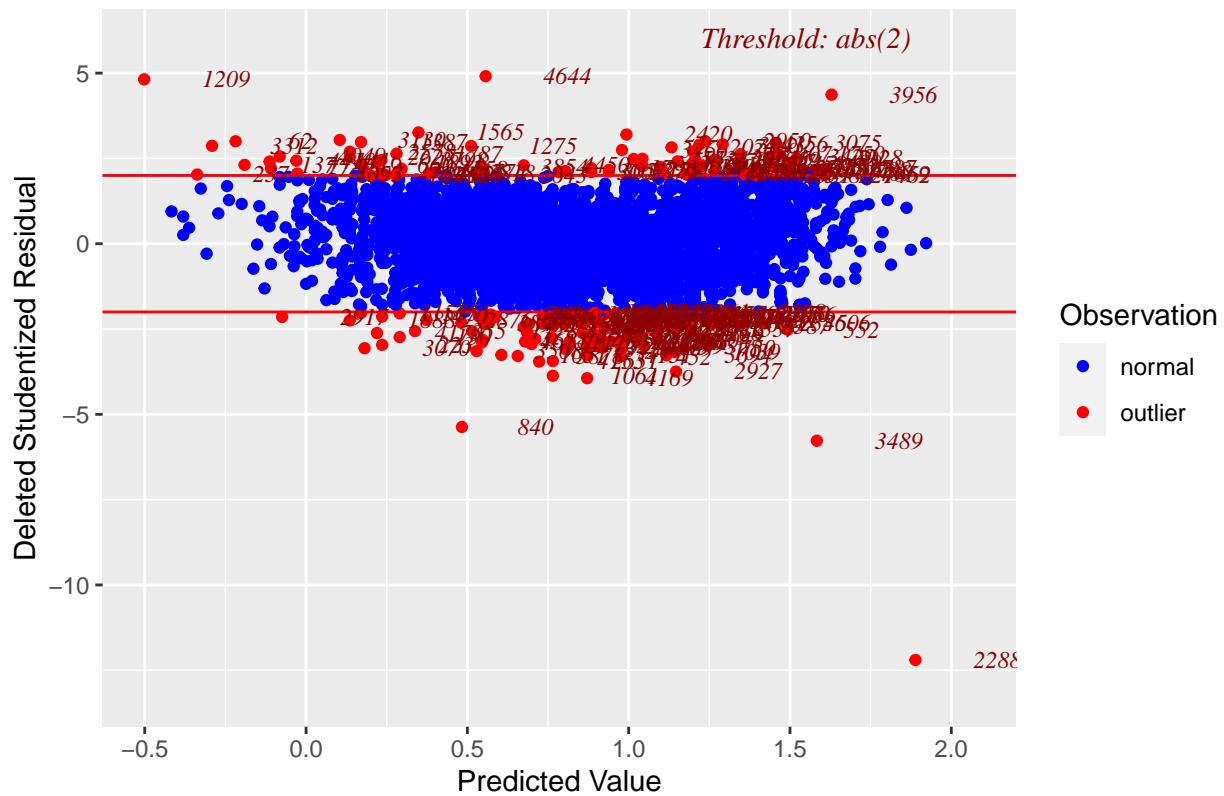
```

stud.del.resids=rstudent(mod2)
range(stud.del.resids) #rango por defecto

## [1] -12.198859 4.908876
n=length(train$Price)
alpha=0.005
valorcritico= qt(1-alpha/(2*n),df=n-3-1) #recordamos que p=k+1 es decir el numero de covariables+1
ols_plot_resid_stud_fit(mod2)

```

Deleted Studentized Residual vs Predicted Values

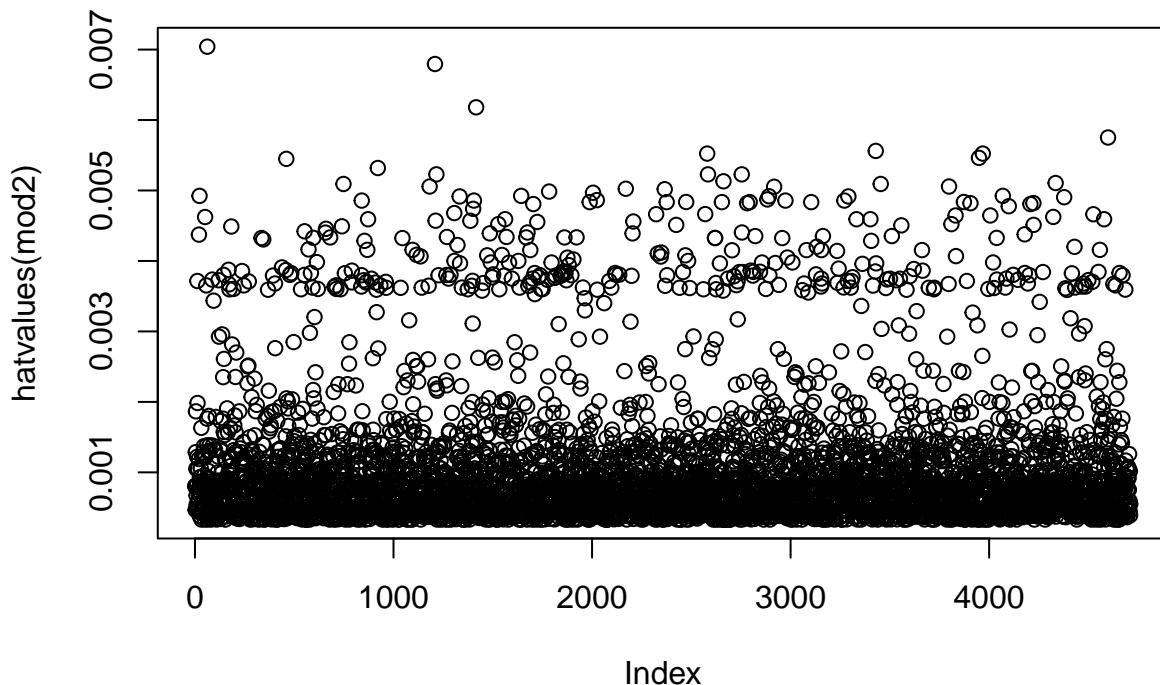


Encontramos multitud de outliers pero la mayoría cercanos a los valores normales. Hay puntos , en concreto 6 de ellos que se alejan bastante de esa normalidad y hacen que el rango de los residuos sea más amplio.

En cuanto a los puntos leva o leverage:

El comando 'hatvalues' nos sirve para obtener directamente los valores h_{ii} o también llamados valores leverage o valores leva.

```
plot(hatvalues(mod2))
```



```
valorLimite = 2*3/n
valorLimite
```

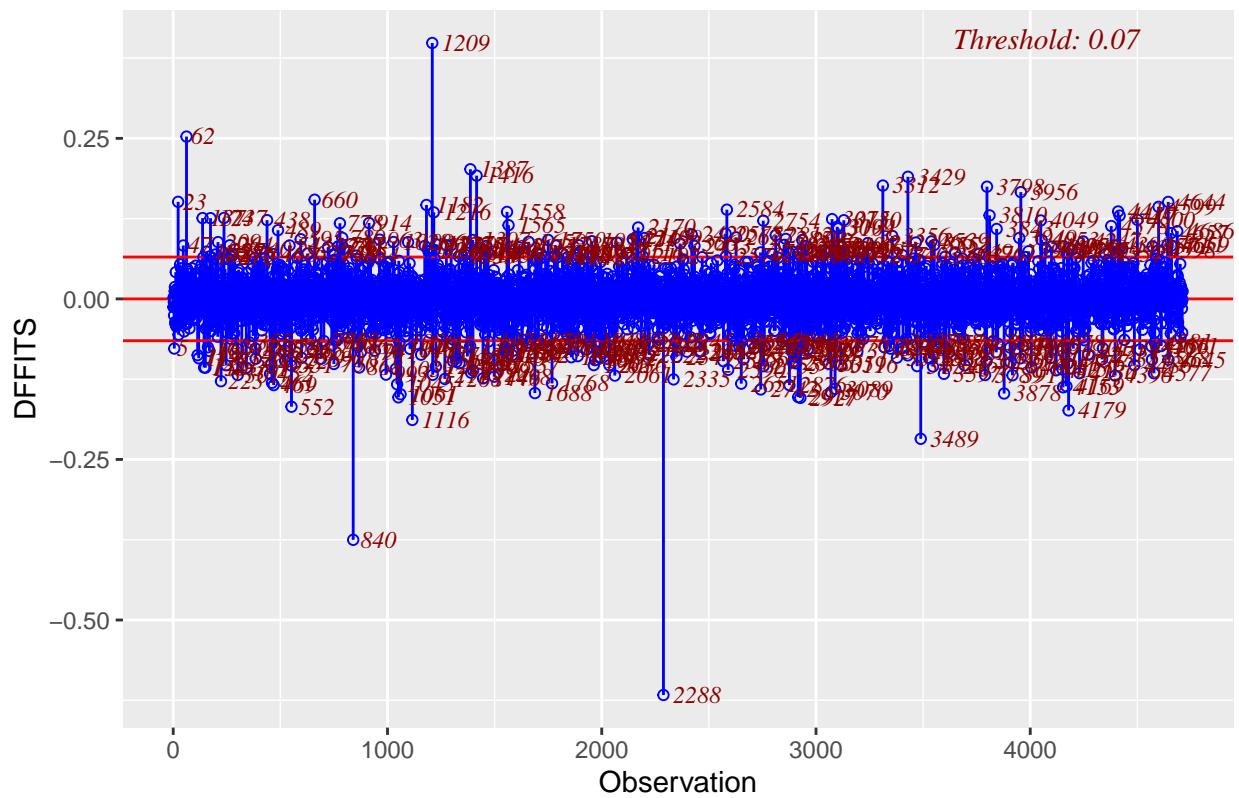
```
## [1] 0.001273615
```

Algunos valores superan el umbral pero la mayoría de las observaciones están concentradas dentro del umbral permitido, por lo que aquí también aparecen varios coches como outliers.

En cuanto a los DFFITS, para el gráfico donde evaluamos si los DFFITS son grandes o pequeños mejor usar como umbrales los valores -1, 1, tal y como comentamos en teoría, cuando analicemos tamaños moderados de data sets. En el caso de analizar grandes bases de datos, **Big data**, está bien la cota que definen por defecto en el gráfico, que usa la formula $2\sqrt{p/n}$. Representan el numero de desviaciones estándar que cambia la predicción de la i-ésima respuesta cuando dicha observación es excluida del ajuste.

```
#dffits(mod2)
ols_plot_dffits(mod2)
```

Influence Diagnostics for $\log_{10}(\text{Price})$

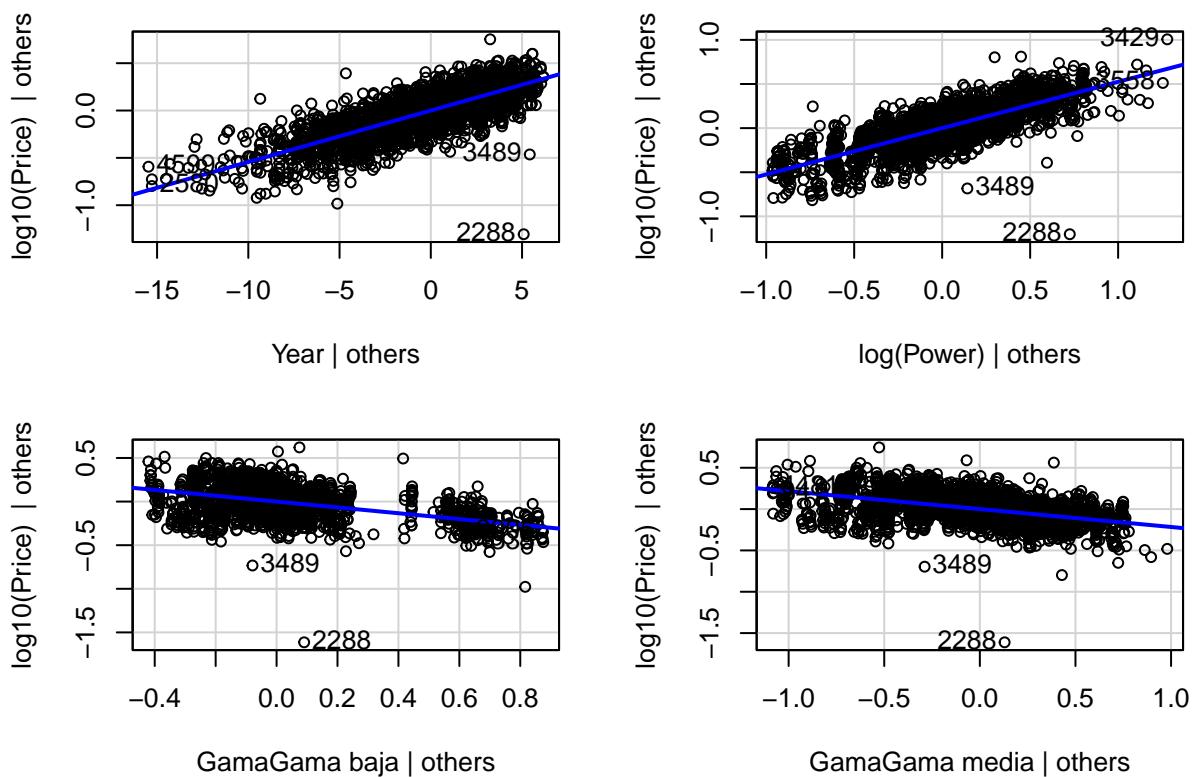


Como en los dos gráficos anteriores cuando analizamos los puntos leverage, aquí también nos aparecen las mismas muestras alejadas de esa media, consideraremos todos esos puntos que están fuera del umbral como observaciones anómalas.

Para ver la linealidad que sigue cada una de las variables regresoras con la variable respuesta, otro gráfico que lo verifica es:

```
avPlots(mod2)
```

Added-Variable Plots



Para analizar más profundamente la variabilidad explicada por el modelo:

```
summary(mod2)
```

```
##
## Call:
## lm(formula = log10(Price) ~ Year + log(Power) + Gama, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.58359 -0.07878  0.00429  0.08516  0.64611 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -1.110e+02  1.215e+00 -91.36 <2e-16 ***
## Year        5.440e-02  6.041e-04  90.05 <2e-16 ***
## log(Power)   5.253e-01  6.239e-03  84.20 <2e-16 ***
## GamaGama baja -3.337e-01  9.763e-03 -34.18 <2e-16 ***
## GamaGama media -2.162e-01  5.742e-03 -37.65 <2e-16 *** 
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.132 on 4706 degrees of freedom
## Multiple R-squared:  0.8773, Adjusted R-squared:  0.8772 
## F-statistic: 8413 on 4 and 4706 DF,  p-value: < 2.2e-16
anova(mod2)

## Analysis of Variance Table
```

```

## 
## Response: log10(Price)
##                               Df Sum Sq Mean Sq  F value    Pr(>F)
## Year                  1 163.22 163.22 9366.85 < 2.2e-16 ***
## log(Power)           1 393.60 393.60 22587.85 < 2.2e-16 ***
## Gama                 2  29.59   14.79   848.91 < 2.2e-16 ***
## Residuals            4706  82.00     0.02
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

La variabilidad del modelo se puede descomponer como $SST = SSM + SSR$. SST es la variación total. SSM es la variación explicada por el modelo. SSR es la variación no explicada por el modelo. Para nuestros datos tenemos: SSM de 163.87,393.63 y 29.58, con SSR 82.03. Por lo tanto, la variabilidad explicada por el modelo de regresión es mayor que la que queda sin explicar (residuos).

Un estadístico F mayor de 1 indica un buen ajuste del modelo. El estadístico F contrasta si el modelo tiene significativa capacidad predictiva. En el contraste la hipótesis nula es $F = 1$, con un p-valor menor de 0.05 ($2.2e-16$) se rechaza la hipótesis nula. Por lo tanto concluimos que el modelo tiene una capacidad predictiva significativa.

PREDICCIÓN SOBRE LOS DATOS DE TESTING. EVALUACIÓN DEL MODELO

Lo primero que tenemos que realizar es la transformación de la variable Make, la cual dividimos en subgrupos. Además, eliminamos las dos columnas que nos eran inútiles del dataset.

```

str_split(test$name, " ", simplify=TRUE) %>% subset(select=1) %>% unique() -> car_makes
car_makes[car_makes=="Land"] <- "Land Rover" #Anomalía en los datos
car_makes_regex <- paste(car_makes, collapse="|")
str_extract(test$name, car_makes_regex) -> make
test %>% mutate(Make = str_to_title(make)) -> test

test %>%
  mutate(
    Gama = case_when(
      test$Make=="Datsun" | test$Make=="Smart" | test$Make=="Tata" | test$Make=="Fiat" | test$Make=="Chevrolet"
      | test$Make=="Skoda" | test$Make=="Renault" | test$Make=="Ford" | test$Make=="Honda" | test$Make=="Volkswagen"
      | test$Make=="Bentley" | test$Make=="Porsche" | test$Make=="Land Rover" | test$Make=="Jaguar" | test$Make=="BMW"
      | test$Make=="Mercedes-Benz" | test$Make=="Audi" | test$Make=="Volvo" | test$Make=="Nissan" | test$Make=="Toyota"
      | test$Make=="Subaru" | test$Make=="Hyundai" | test$Make=="Kia" | test$Make=="Mazda" | test$Make=="Ford"
      | test$Make=="Dodge" | test$Make=="Chrysler" | test$Make=="Jeep" | test$Make=="BMW" | test$Make=="Vauxhall"
      | test$Make=="Citroen" | test$Make=="Peugeot" | test$Make=="Renault" | test$Make=="VW" | test$Make=="VW"
      | test$Make=="Vauxhall" | test$Make=="Vauxhall" | test$Make=="Vauxhall" | test$Make=="Vauxhall"
      )
    ) -> test

test$Gama <- as.factor(test$Gama)

test %>% select(-New_Price) -> test
test %>% select(-X) -> test

```

Y para finalizar, veamos cómo predice el modelo:

```

pred=predict(mod2,test[,-12])
tables=as.data.frame(cbind(Prediccion=exp(pred),Observ=log(test[,12]), Dif=exp(pred)-log(test[,12])))
summary(tables)

##      Prediccion          Observ            Dif
## Min.   :0.5911   Min.   :-0.7985   Min.   :-0.8596
## 1st Qu.:1.7978   1st Qu.: 1.2733   1st Qu.: 0.3005
## Median :2.1494   Median : 1.7317   Median : 0.4931

```

```

##  Mean    :2.3905   Mean    : 1.8498   Mean    : 0.5228
##  3rd Qu.:2.7873   3rd Qu.: 2.3026   3rd Qu.: 0.6915
##  Max.    :5.9959   Max.    : 4.7875   Max.    : 2.4125
##  NA's    :31        NA's    :31        NA's    :31

```

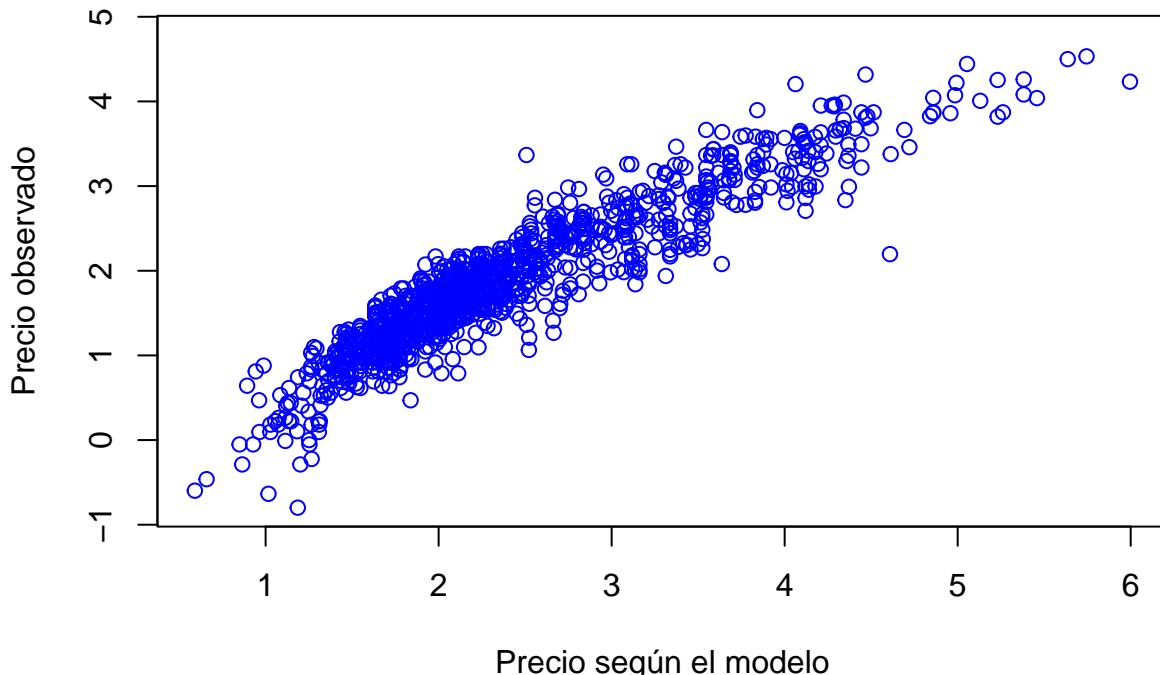
Debido a que tenemos un error pequeño de predicción podemos considerar que tenemos un buen modelo.

```

plot(exp(pred),log(test$Price),col="blue", xlab = "Precio según el modelo",
      ylab = "Precio observado", main="Representación del precio predicho frente al observado")

```

Representación del precio predicho frente al observado



```
#Análisis de componentes principales
```

El análisis de componentes principales (en inglés, PCA) es una técnica utilizada para describir un conjunto de datos en términos de nuevas variables denominadas componentes no correlacionadas. Estas nuevas componentes se construyen a partir de las variables existentes, eso sí, debemos asegurarnos de que las variables utilizadas en PCA sean variables cuantitativas (no podemos usar variables cualitativas ni categóricas). Con esta técnica se pretende reducir la dimensionalidad del problema en cuestión.

```
str(train)
```

```

## #tibble [4,711 x 16] (S3:tbl_df/tbl/data.frame)
## $ Name          : chr [1:4711] "Audi Q3 2012-2015 35 TDI Quattro Premium" "BMW 3 Series 320d Lux"
## $ Location       : Factor w/ 11 levels "Ahmedabad","Bangalore",...: 10 10 3 8 11 5 10 4 10 3 ...
## $ Year          : int [1:4711] 2015 2014 2016 2009 2005 2016 2017 2011 2012 ...
## $ Kilometers_Driven: int [1:4711] 48000 18600 18000 80464 123200 46727 17000 30090 32000 115000 ...
## $ Fuel_Type      : Factor w/ 5 levels "CNG","Diesel",...: 2 2 5 2 5 2 5 2 5 2 ...
## $ Transmission   : Factor w/ 2 levels "Automatic","Manual": 1 1 1 1 2 2 1 1 2 1 ...
## $ Owner_Type     : Factor w/ 4 levels "First","Second",...: 1 2 1 2 2 1 1 1 2 3 ...
## $ Engine          : num [1:4711] 1968 1995 1197 2987 1495 ...
## $ Power           : num [1:4711] 174 190 82 198 94 ...
## $ Seats            : int [1:4711] 5 5 5 5 5 5 5 5 5 ...
## $ Price            : num [1:4711] 18.25 21 5.4 9.29 1 ...

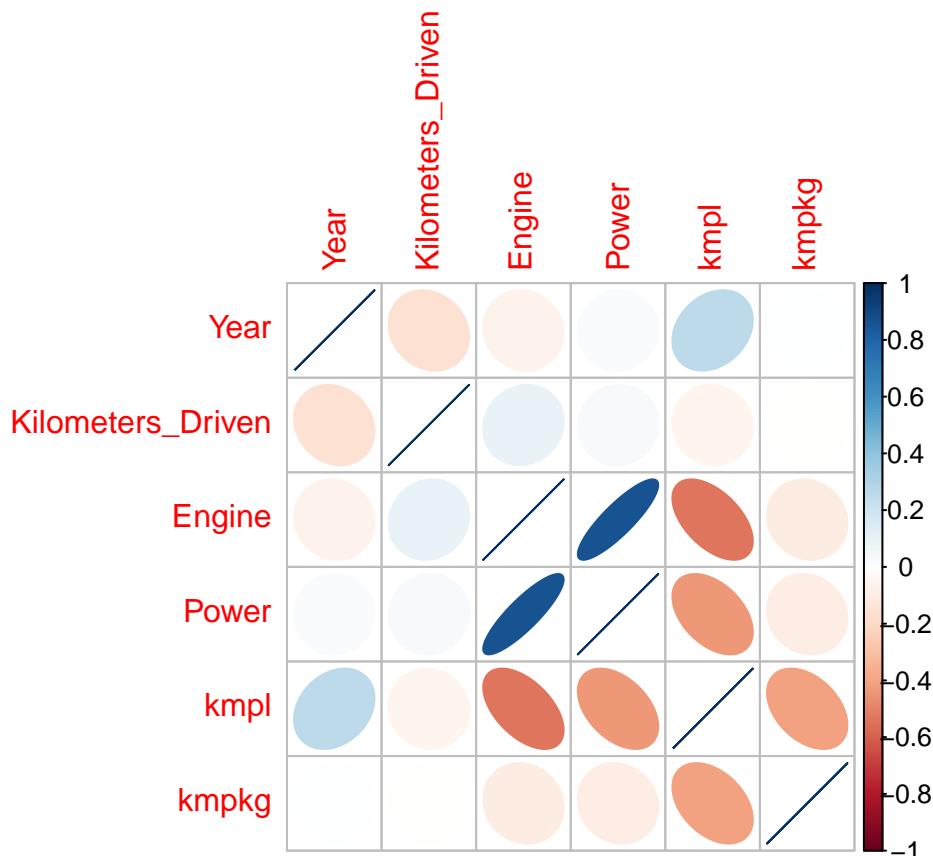
```

```

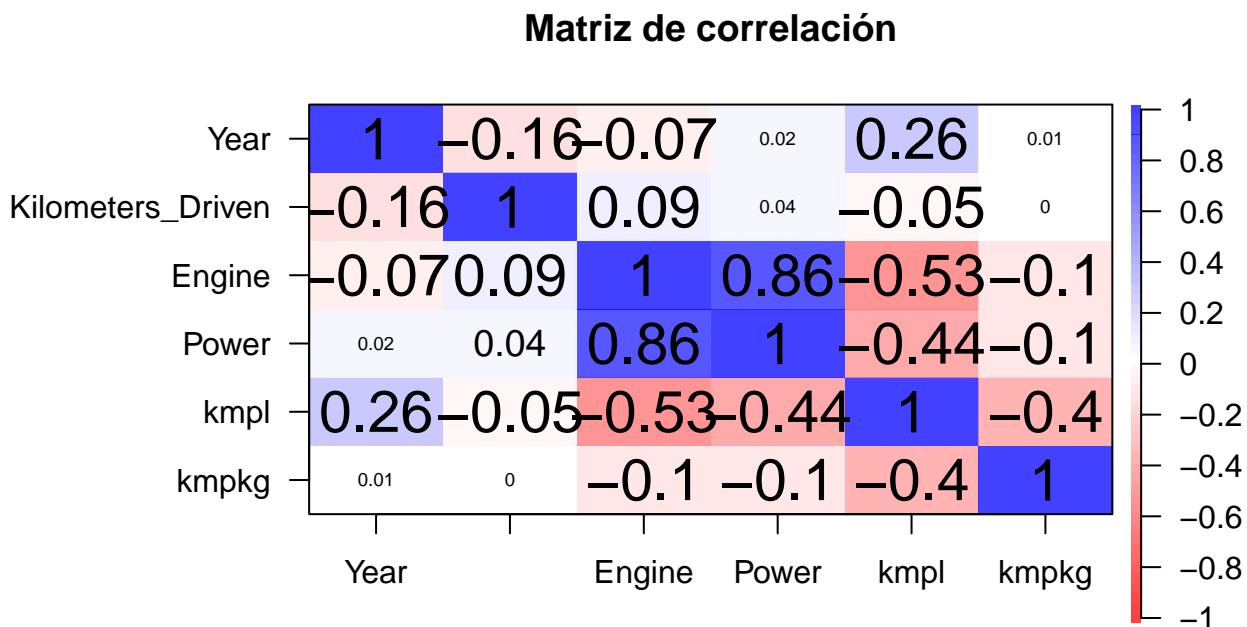
## $ Make          : chr [1:4711] "Audi" "Bmw" "Hyundai" "Mercedes-Benz" ...
## $ Gama          : Factor w/ 3 levels "Gama alta","Gama baja",... 1 1 3 1 3 3 1 3 3 1 ...
## $ kmpl          : num [1:4711] 15.7 22.7 18.9 11 13.2 ...
## $ kmpkg          : num [1:4711] 0 0 0 0 0 0 0 0 0 0 ...
## $ Price.mod2    : Named num [1:4711] 1.32 1.311 0.763 1.062 0.236 ...
## ..- attr(*, "names")= chr [1:4711] "1" "2" "3" "4" ...
train

## # A tibble: 4,711 x 16
##   Name      Location Year Kilometers_Driv~ Fuel_Type Transmission Owner_Type
##   <chr>     <fct>    <int>        <int> <fct>       <fct>       <fct>
## 1 Audi Q3 20~ Mumbai    2015        48000 Diesel Automatic First
## 2 BMW 3 Seri~ Mumbai    2014        18600 Diesel Automatic Second
## 3 Hyundai Gr~ Chennai   2016        18000 Petrol Automatic First
## 4 Mercedes-B~ Kochi     2009        80464 Diesel Automatic Second
## 5 Hyundai Ac~ Pune      2005       123200 Petrol Manual  Second
## 6 Ford EcoSp~ Delhi     2016        46727 Diesel Manual  First
## 7 Mercedes-B~ Mumbai    2016        17000 Petrol Automatic First
## 8 Renault Du~ Coimbat~  2017        30090 Diesel Automatic First
## 9 Ford Figo ~ Mumbai    2011        32000 Petrol Manual  Second
## 10 BMW X1 sDr~ Chennai   2012       115000 Diesel Automatic Third
## # ... with 4,701 more rows, and 9 more variables: Engine <dbl>, Power <dbl>,
## #   Seats <int>, Price <dbl>, Make <chr>, Gama <fct>, kmpl <dbl>, kmpkg <dbl>,
## #   Price.mod2 <dbl>
corrplot(cor(train[, c(3,4, 8, 9, 14, 15)]), method = "ellipse")

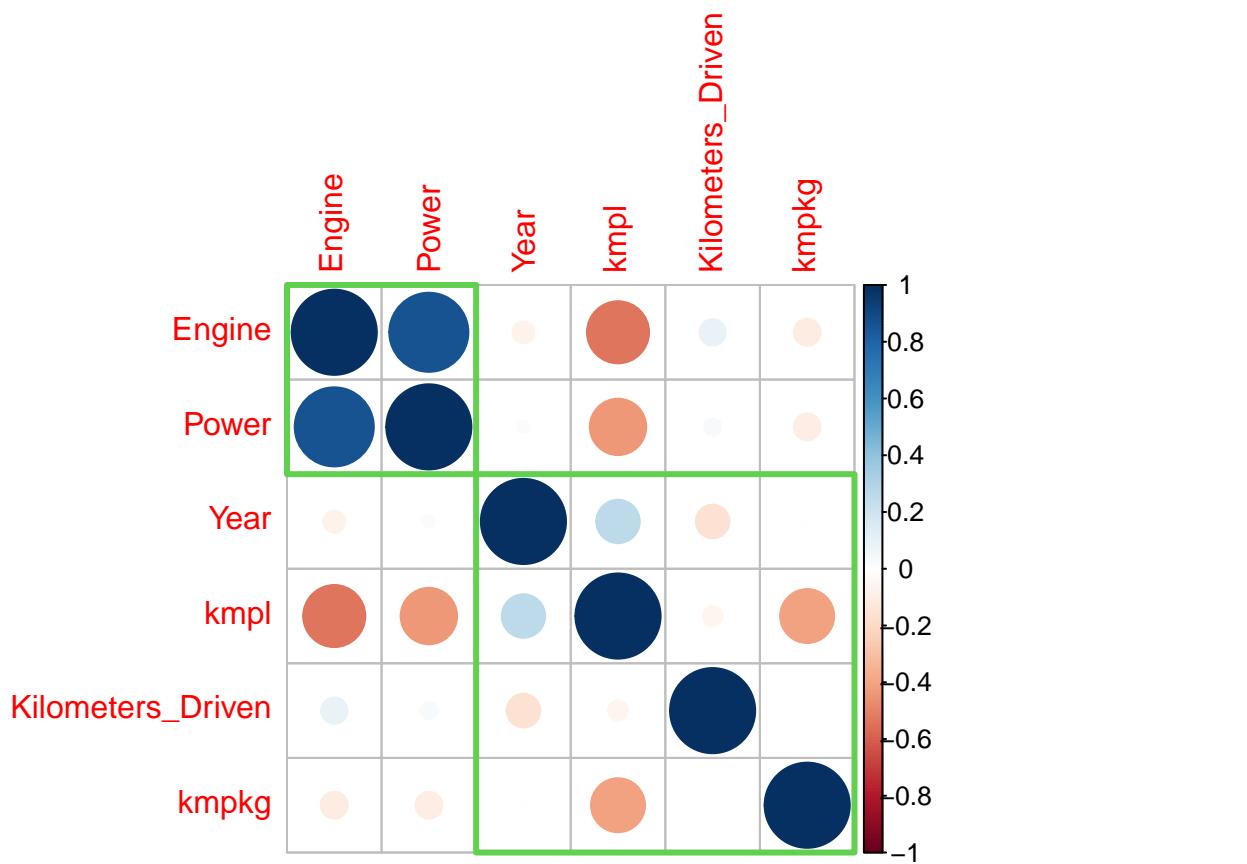
```



```
corPlot(train[, c(3,4, 8, 9, 14, 15)], cex = 1.2, main = "Matriz de correlación")
```



```
corrplot(cor(train[, c(3,4, 8, 9, 14, 15)]), method = "circle", order = "hclust",  
addrect = 2, rect.col = 3, rect.lwd = 3)
```



```
cortest(cor(train[, c(3,4, 8, 9, 14, 15)]))
```

```

## Warning in cortest(cor(train[, c(3, 4, 8, 9, 14, 15)])): n not specified, 100
## used

## Tests of correlation matrices
## Call:cortest(R1 = cor(train[, c(3, 4, 8, 9, 14, 15)]))
## Chi Square value 250.68 with df = 15 with probability < 9e-45

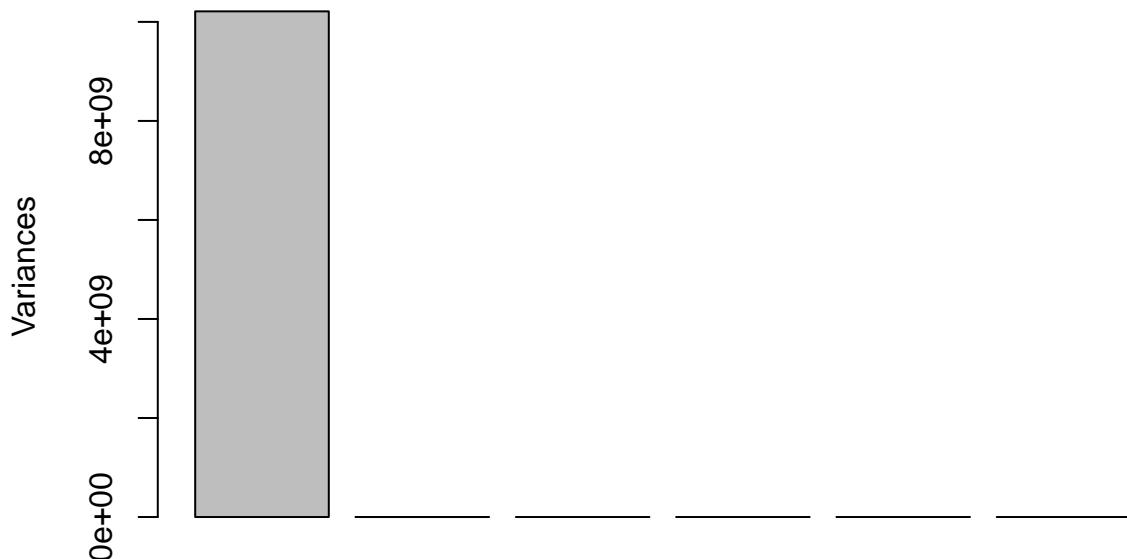
```

Tenemos evidencias para decir que las correlaciones son distintas de 0

```
pca1 <- prcomp(train[, c(3,4, 8, 9, 14, 15)])
```

```
plot(pca1)
```

pca1



```
summary(pca1)
```

```

## Importance of components:
##              PC1       PC2       PC3       PC4       PC5       PC6
## Standard deviation   101058 600.56073 27.03 4.409 3.046 1.968
## Proportion of Variance    1  0.00004  0.00 0.000 0.000 0.000
## Cumulative Proportion    1  1.00000  1.00 1.000 1.000 1.000

```

En nuestro conjunto de datos inicial, todas nuestras variables eran cuantitativas (menos la variable respuesta, que no utilizamos en aprendizaje no supervisado), sin embargo, las variables categorizadas que hemos creado no lo son. Así que haremos el análisis de componentes principales escalando las variables: Kilometers_Driven, Power, Seats, kmpl y kmpg.

```
pca2 <- prcomp(train[, c(3,4, 8, 9, 14, 15)], scale=T)
```

```
pca2
```

```

## Standard deviations (1, ..., p=6):
## [1] 1.5099667 1.1444561 1.0599891 0.9273837 0.5500460 0.3522111
##
## Rotation (n x k) = (6 x 6):
##              PC1          PC2          PC3          PC4          PC5
## Year      -0.13713864 -0.3782024  0.596223411  0.63439205  0.28287926
## Kilometers_Driven  0.10115280  0.1465697 -0.668955170  0.71929106  0.04377908

```

```

## Engine          0.61114224 -0.2395341  0.007784461 0.01713324 -0.11602069
## Power          0.58101040 -0.3074207  0.085329519 0.04975366 -0.38891068
## kmpl           -0.50571711 -0.4201464 -0.135355634 0.08348424 -0.71719694
## kmpkg          0.06436916  0.7120907  0.413948973 0.26538370 -0.48885474
##               PC6
## Year            -0.01412815
## Kilometers_Driven 0.03857703
## Engine          -0.74519368
## Power           0.63789605
## kmpl            -0.16752599
## kmpkg           -0.08956703

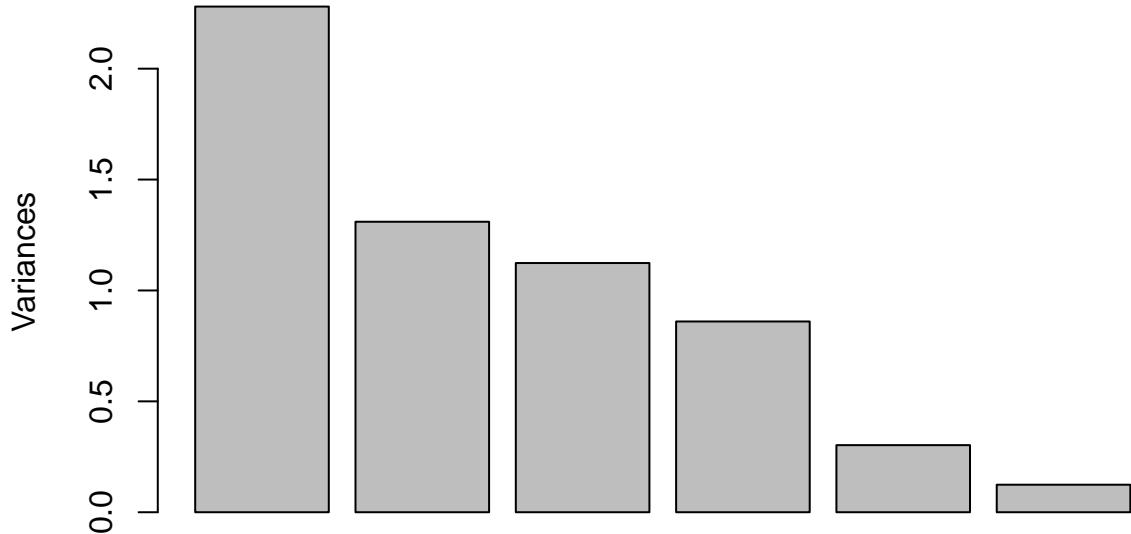
```

Aquí arriba, podemos ver los diferentes pesos que otorga el análisis de componentes principales a cada una de las variables iniciales escaladas. Por ejemplo: en la primera componente principal (PC1), vemos que sobre todo se enfrentan las variables Engine y Power contra kmpl; en la segunda componente principal (PC2), vemos que se enfrenta la variable kmpg contra kmpl, Power, Year y Engine; en la tercera componente principal (PC3) se enfrentan los kilómetros que lleva recorridos el coche contra el año de fabricación y la variable kmpkg.

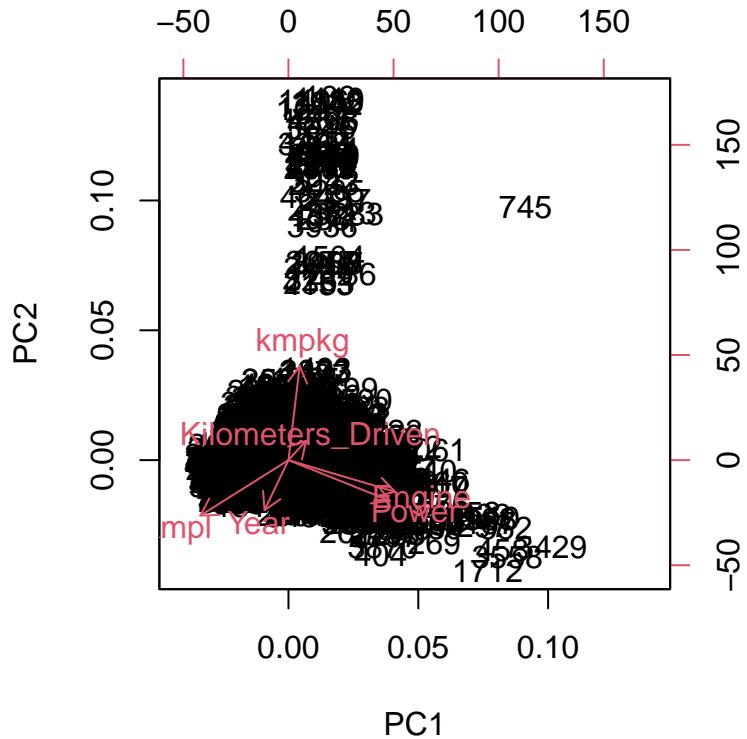
Sin embargo, aquí resulta difícil ver algo claro e intuitivo, así que haremos un pequeño resumen y un gráfico multivariante para mostrar la información más relevante del PCA.

```
plot(pca2)
```

pca2



```
biplot(pca2)
```

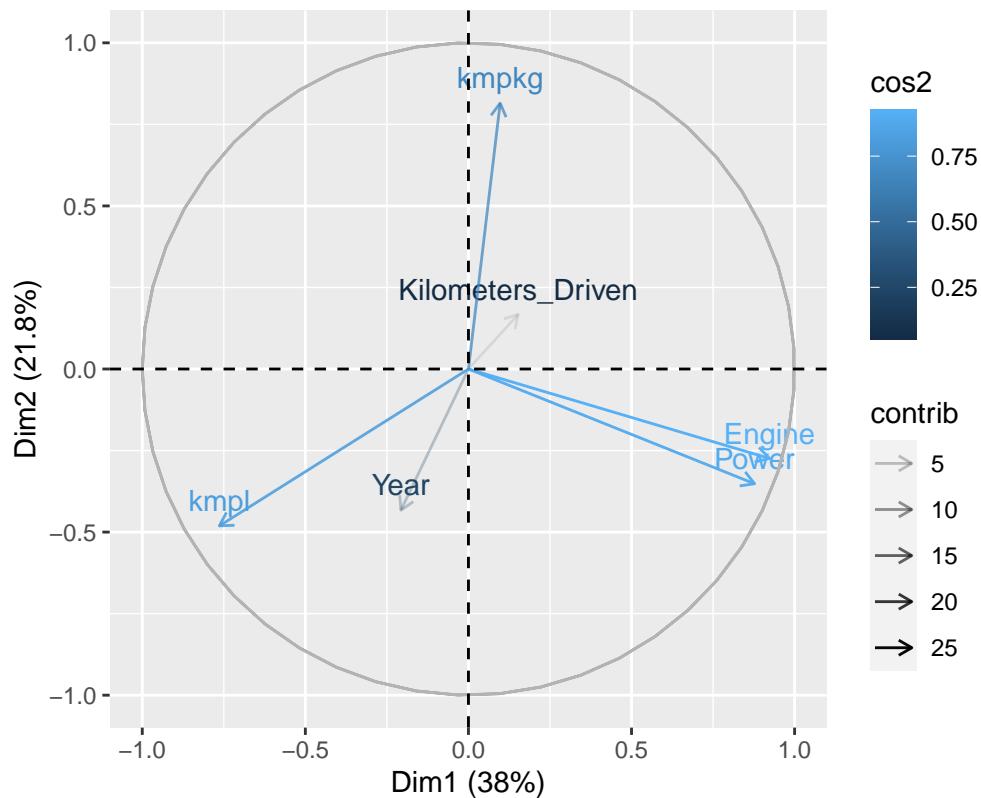


En el gráfico enfrentamos la primera y la segunda componente principal, y vemos como influyen cada una de las variables en los coches. Por ejemplo, el coche 1712, debe tener unos valores muy altos de Power y Engine que son las variables que “más tiran hacia la derecha”, y el teorema 745, según el biplot debe tener un valor muy alto de kilometers_driven, que es la variable que “más tira en esa dirección”. Si recordamos lo analizado previamente, en la sección del análisis exploratorio de datos, este dos teoremas ya destacó por tener valores un tanto diferentes a los del resto por su desmesurado valor de kilómetros recorridos. Comprobamos que la información que nos proporciona el gráfico es totalmente coherente con lo que obtuvimos en el *EDA*.

```
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
fviz_pca_var( pca2, axes = c( 1, 2), col.var = "cos2", alpha.var = "contrib" ) + theme_grey()
```

Variables – PCA

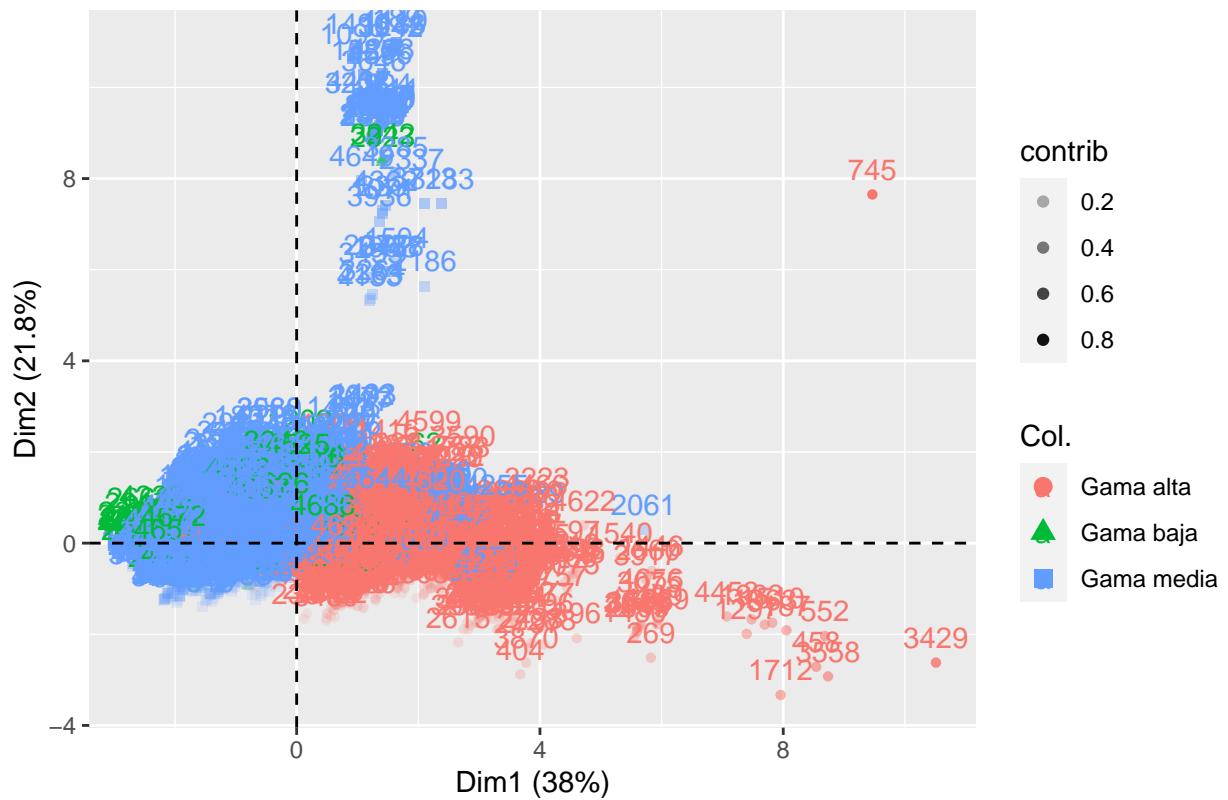


La suma de cos² de una variable determinada sobre cada factor es 1. Esto significa que cada vector debería estar tocando el perímetro de la circunferencia unidad, pero no lo está haciendo prácticamente, ¿por qué?. Si observamos por ejemplo la variable Engine(al igual que Power), vemos que está muy cerca de tocar dicho perímetro, su proyección sobre las dimensiones 1 y 2 (componentes) indica su contribución a éstas, pero aún le falta algo de contribución que debe estar repartida por otra u otras dimensiones. Si esta variable solo tuviese peso sobre las dos primeras dimensiones estaría tocando la circunferencia.

Podemos colorear las observaciones según alguna variable. Además podemos hacer que las variables que más contribuyen en este plano factorial, se resalten más que las que menos influencia tienen. También tenemos la posibilidad de dibujar elipses alrededor de cada grupo con un cierto nivel de confianza.

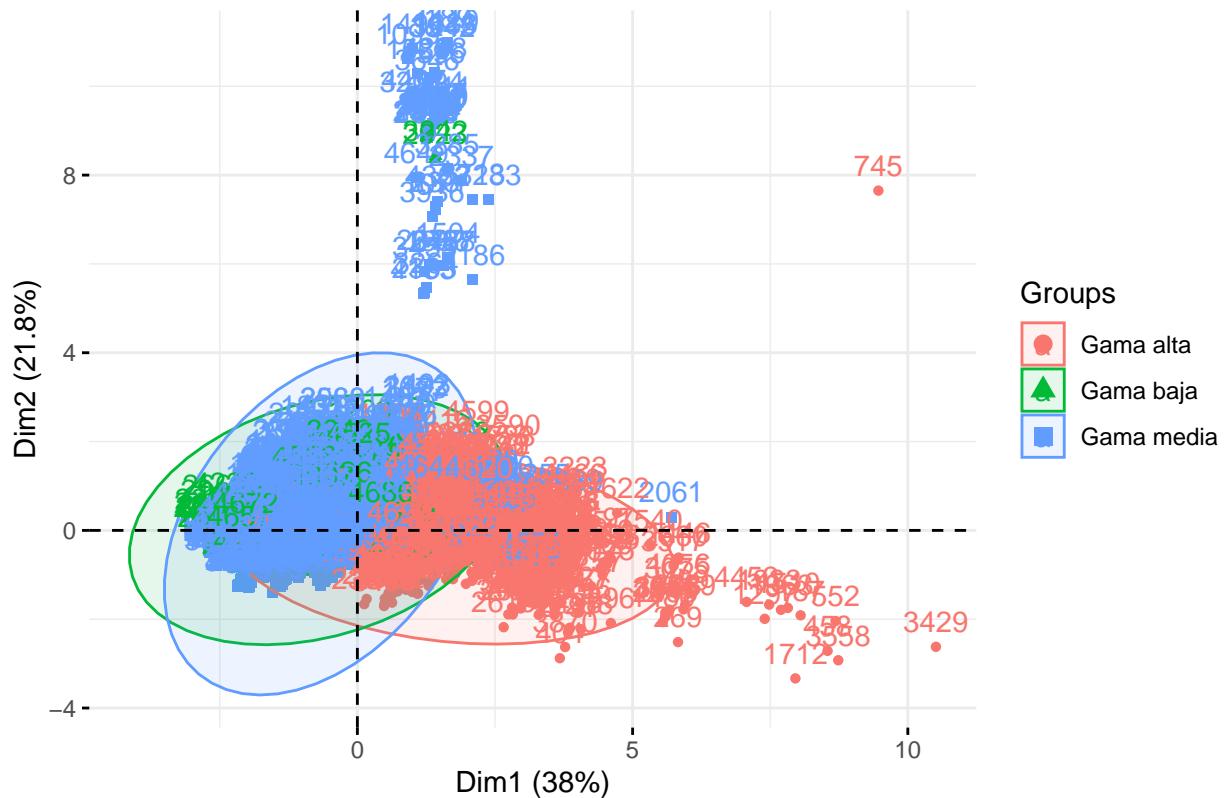
```
fviz_pca_ind( pca2, col.ind = train$Gama, alpha.ind = "contrib" ) + theme_grey()
```

Individuals – PCA



```
fviz_pca_ind( pca2, habillage = as.factor( train$Gama ), addEllipses = TRUE, ellipse.level = 0.99 )
```

Individuals – PCA



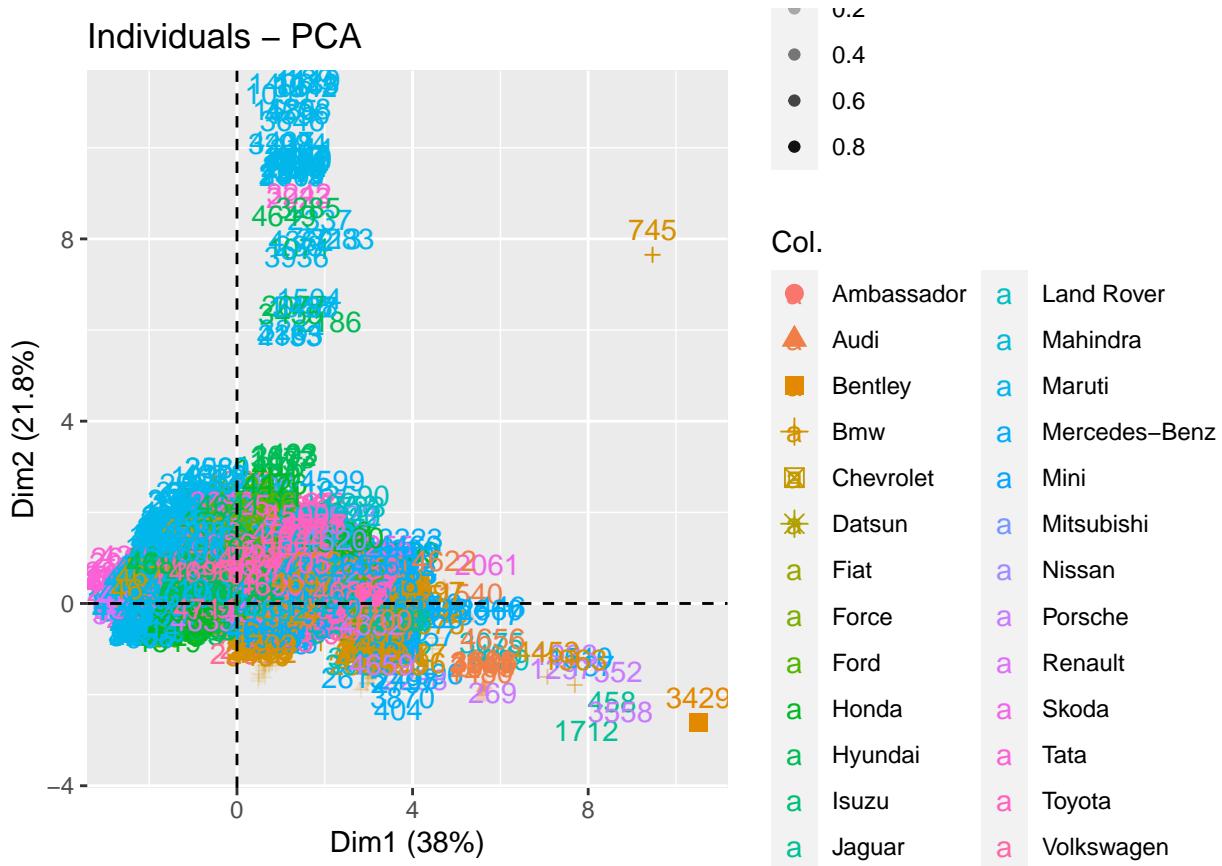
```
fviz_pca_ind( pca2, col.ind = train$Make, alpha.ind = "contrib" ) + theme_grey()
```

```
## Warning: The shape palette can deal with a maximum of 6 discrete values because
## more than 6 becomes difficult to discriminate; you have 28. Consider
## specifying shapes manually if you must have them.
```

```
## Warning: Removed 4207 rows containing missing values (geom_point).
```

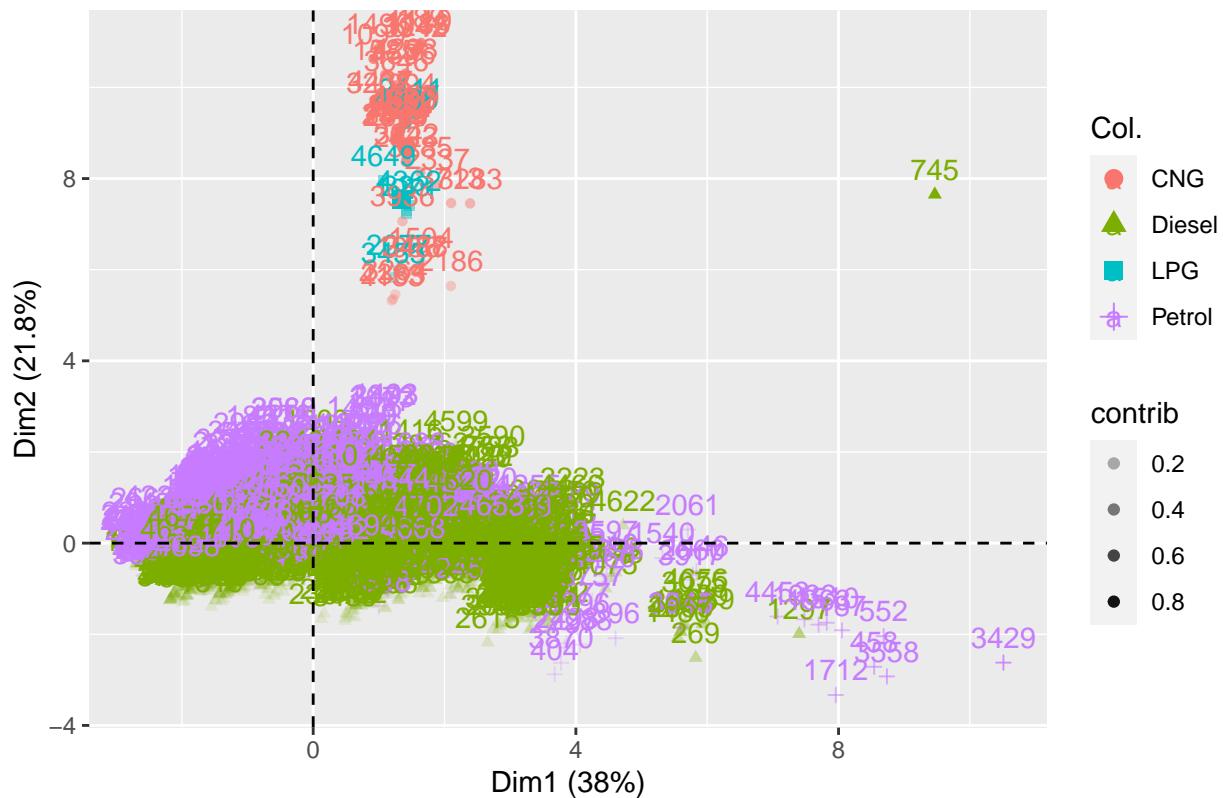
```
## Warning: Removed 22 rows containing missing values (geom_point).
```

Individuals – PCA



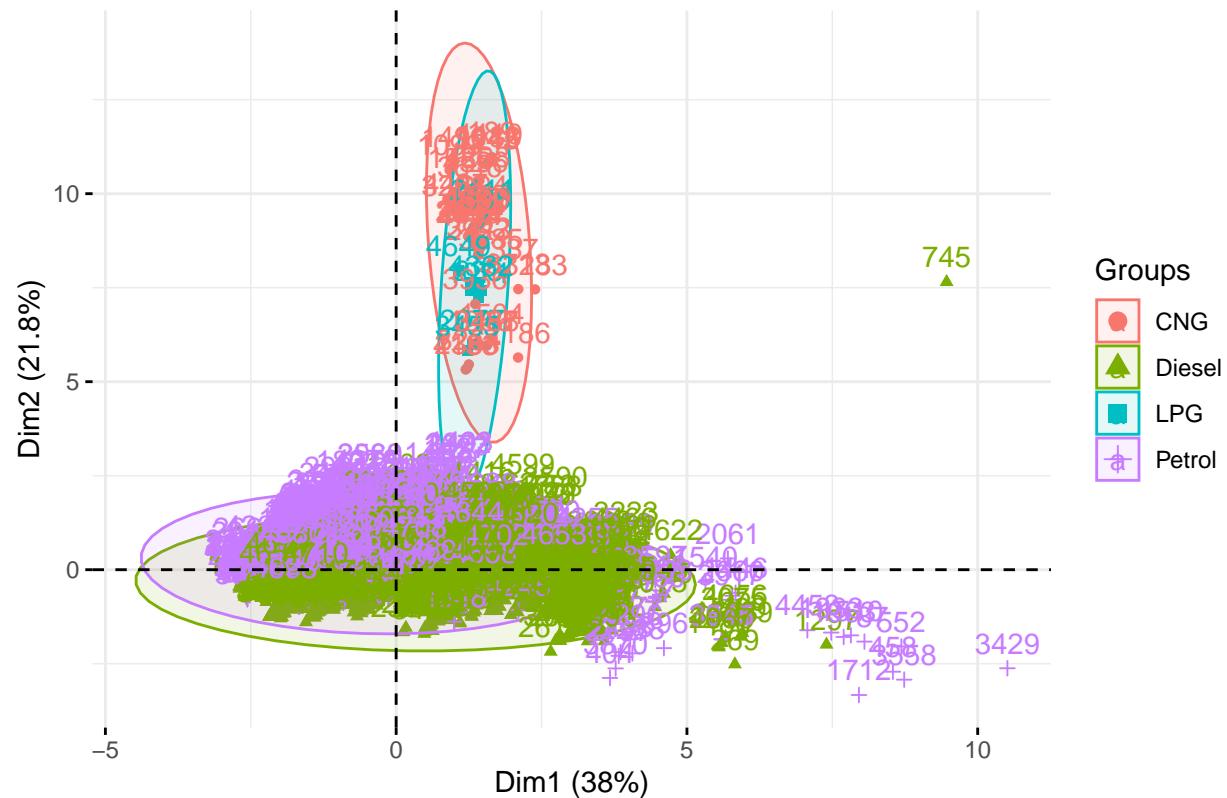
```
fviz_pca_ind( pca2, col.ind = train$Fuel_Type, alpha.ind = "contrib" ) + theme_grey()
```

Individuals – PCA



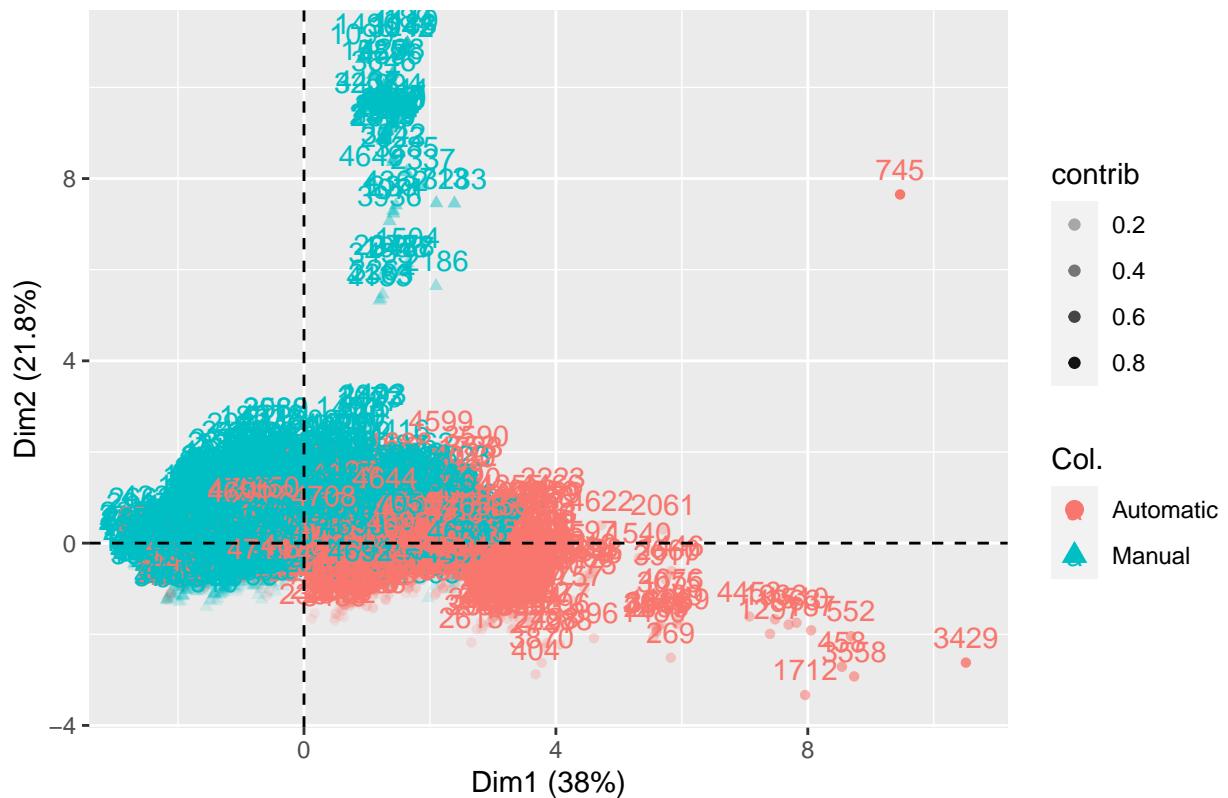
```
fviz_pca_ind( pca2, habillage = as.factor( train$Fuel_Type ), addEllipses = TRUE, ellipse.level = 0.99 )
```

Individuals – PCA



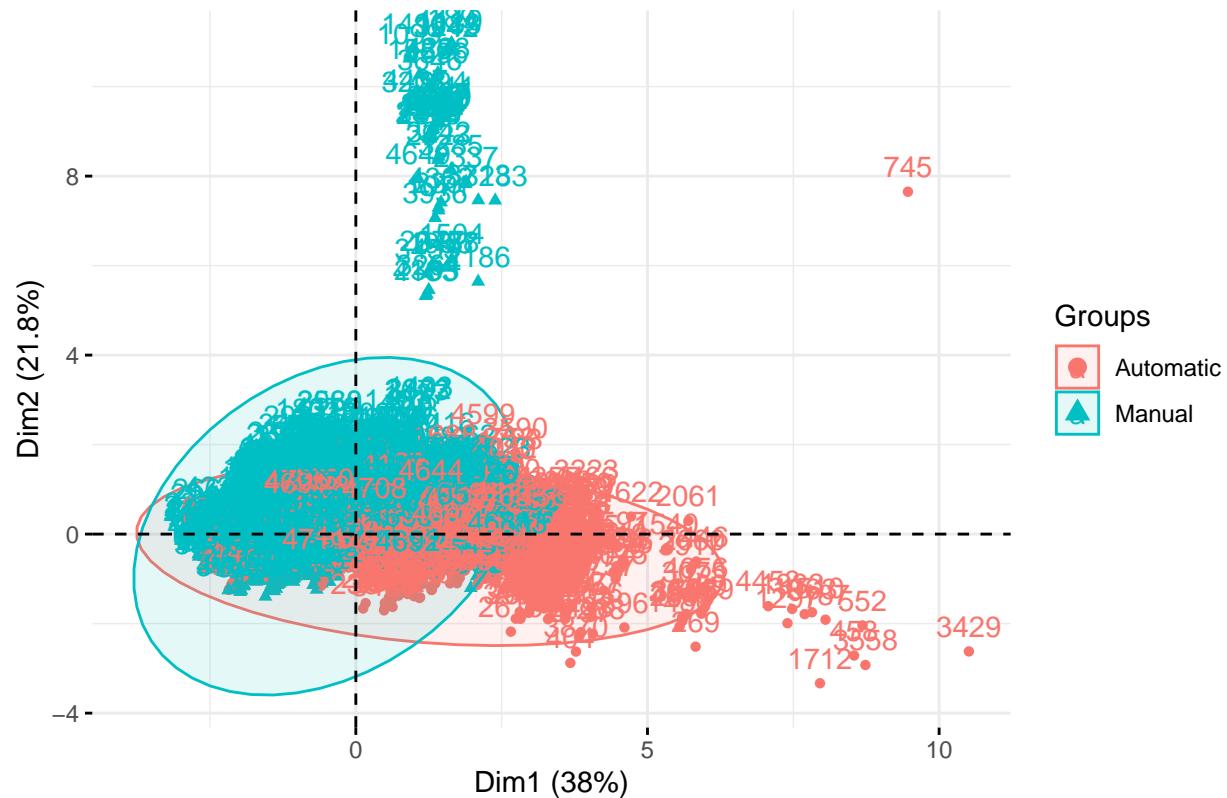
```
fviz_pca_ind( pca2, col.ind = train$Transmission, alpha.ind = "contrib" ) + theme_grey()
```

Individuals – PCA

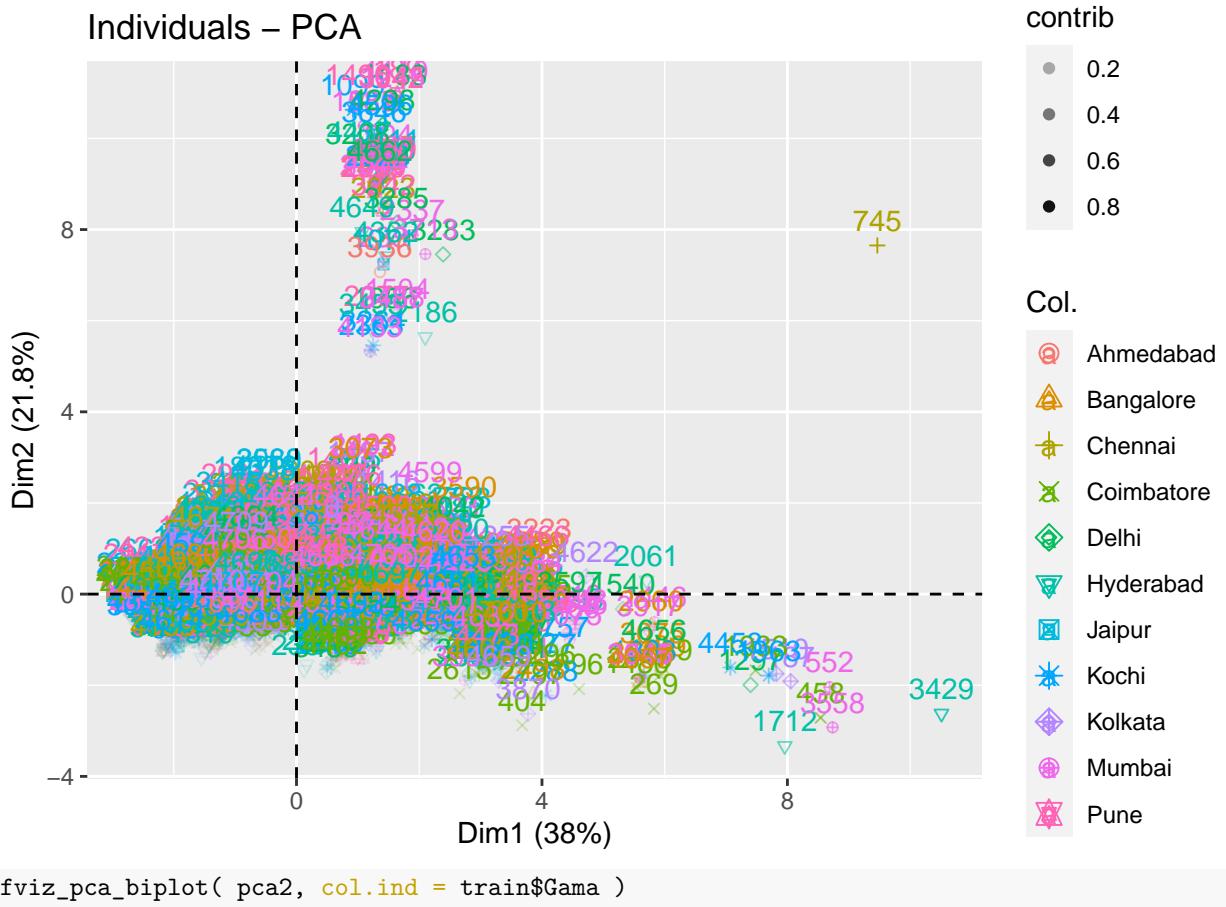


```
fviz_pca_ind( pca2, habillage = as.factor( train$Transmission ), addEllipses = TRUE, ellipse.level = 0.95 )
```

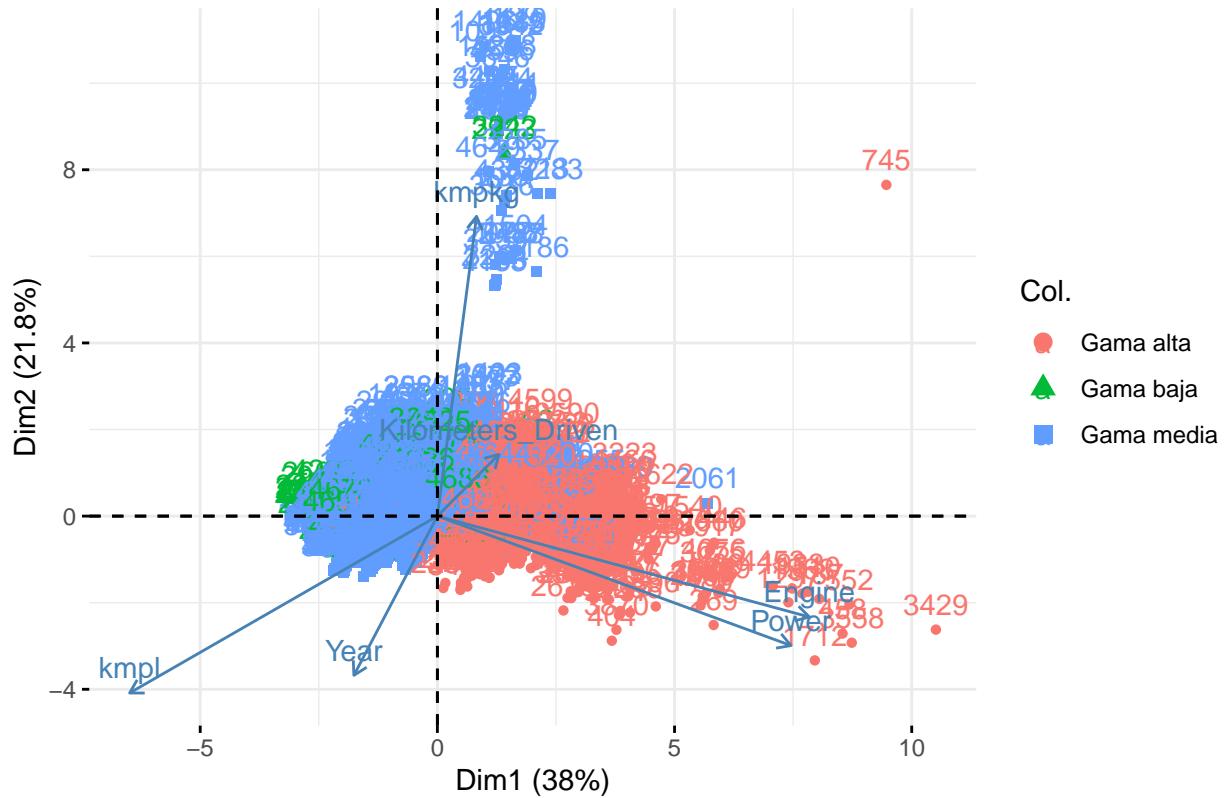
Individuals – PCA



Individuals – PCA



PCA – Biplot



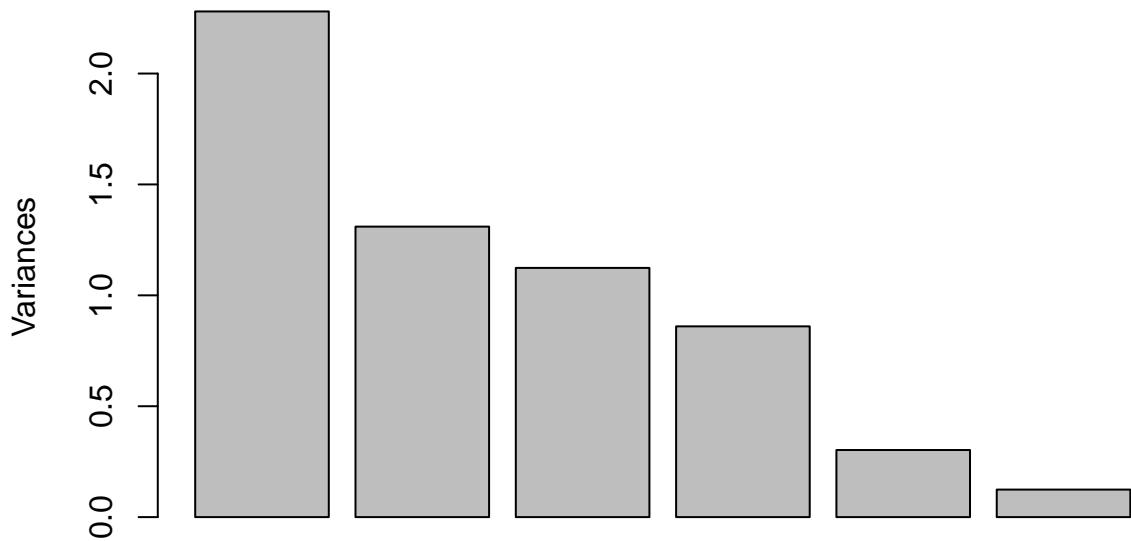
```
summary(pca2)
```

```
## Importance of components:
##              PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation   1.51  1.1445  1.0600  0.9274  0.55005  0.35221
## Proportion of Variance 0.38  0.2183  0.1873  0.1433  0.05043  0.02068
## Cumulative Proportion 0.38  0.5983  0.7856  0.9289  0.97932  1.00000
```

Por otro lado, podemos decir que lo que más nos interesa de este resumen es la proporción de la varianza total que consigue explicar cada componente principal. Según el resumen que acabamos de mostrar arriba, vemos que la varianza total explicada no aumenta mucho a partir de la tercera o cuarta componente principal (y que con todas las componentes principales, evidentemente, la varianza explicada es el 100%). Para visualizar esto haremos un gráfico de barras:

```
screeplot(pca2, xlab="PCs")
```

pca2



PCs

A la luz de

los datos visuales recabados parece evidente que nos tenemos que quedar con las 3 primeras componentes principales (PC1, PC2 y PC3), pero no está tan claro que debemos hacer con la cuarta(PC4).

```
library(psych)
scree(train[, c(3,4, 8, 9, 14, 15)],main ="Grafico_de_Sedimentacion")

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, :
## An
## ultra-Heywood case was detected. Examine the results carefully
```

Grafico_de_Sedimentacion

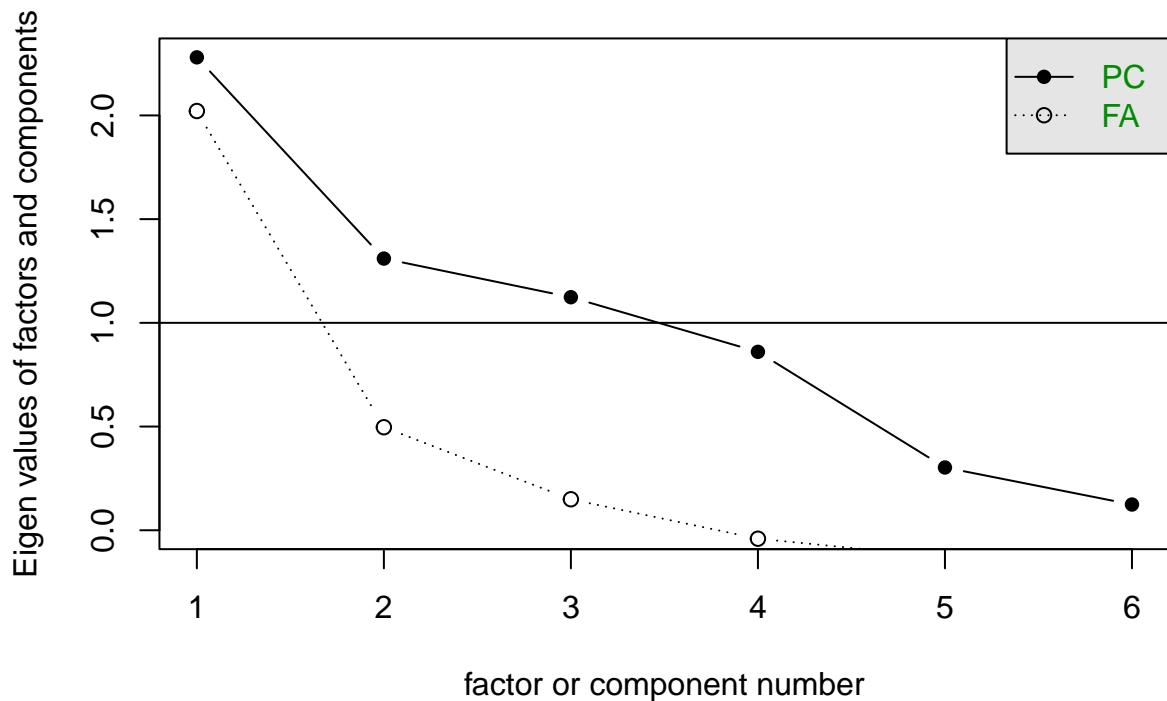


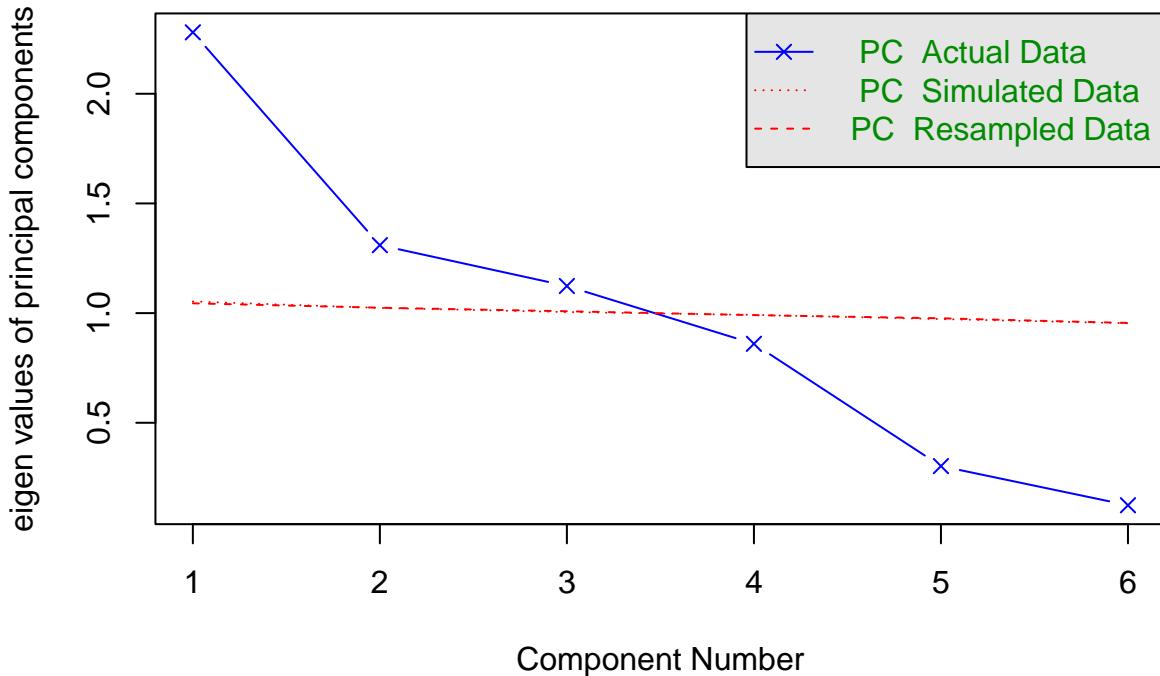
grafico de Sedimentacion nos muestra la cantidad óptima de componentes a tomar en el análisis, siendo los valores por encima de la linea de 1.0 los más aceptables.

```
fa.parallel(train[, c(3,4, 8, 9, 14, 15)], fa="pc")

## Warning in fa.stats(r = r, f = f, phi = phi, n.obs = n.obs, np.obs = np.obs, :
## The estimated weights for the factor scores are probably incorrect. Try a
## different factor score estimation method.

## Warning in fac(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate, :
## An
## ultra-Heywood case was detected. Examine the results carefully
```

Parallel Analysis Scree Plots



```
## Parallel analysis suggests that the number of factors = NA and the number of components = 3
```

Según los resultados del Análisis paralelo, el número de componentes deberá ser 3.