241012_R-Básico_4

J.Ballesta

20/10/24

Resumen

En esta publicación seguiremos con las posibilidades de gráficos con ggplot2 y el uso en su forma más sencilla de la regresión lineal tanto simple como polinómica.

In this post, we will continue exploring the possibilities of graphs with ggplot2 and the use of simple and polynomial linear regression in its most basic form.

In diesem Beitrag werden wir die Möglichkeiten von Diagrammen mit ggplot2 sowie die Anwendung der einfachen und polynomialen linearen Regression in ihrer einfachsten Form fortsetzen.

Introducción.

En esta publicación presentaremos resumidamente las alternativas de regresión lineal que nos ofrece R, tanto en el caso de dos variables como varias variables.

Packages.

En primer lugar, cargamos las librerías que vamos a necesitar en esta sesión:

```
#
#    cargo las librerías necesarias para esta publicación
library (magrittr)       # para poder usar el operador pipeline %>%
library (ggplot2)        # presentación de gráficos
theme_set(theme_bw())    # fijo por defecto el tema de los gráficos en blanco y negro
library(scales)          # ajustes de formatos en los ejes de los gráficos
library (GGally)         # gráficos de Trellis para la regresión polinómica
library(DataExplorer)    # análisis exploratorio de datos EDA
#
```

Conjuntos de datos (*datasets*) para este estudio.

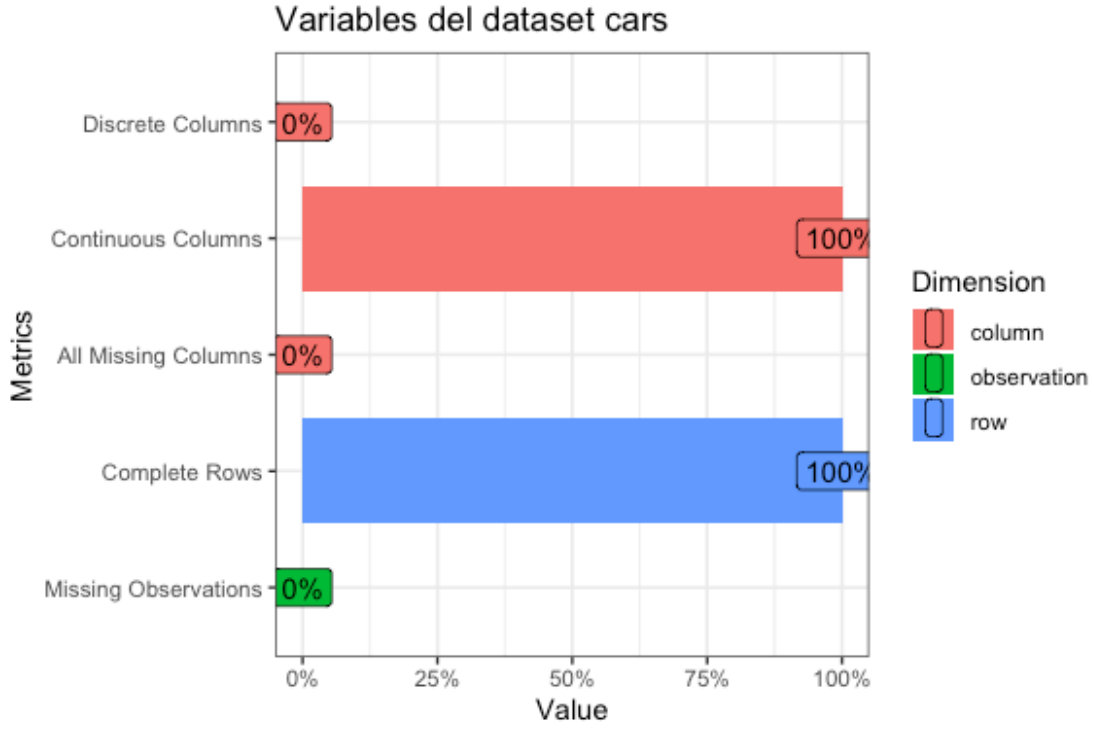
Regresión lineal simple.

En este caso estudiamos la relación entre dos variables, elegimos para esta publicación el dataset *cars*, que contiene las siguientes variables :

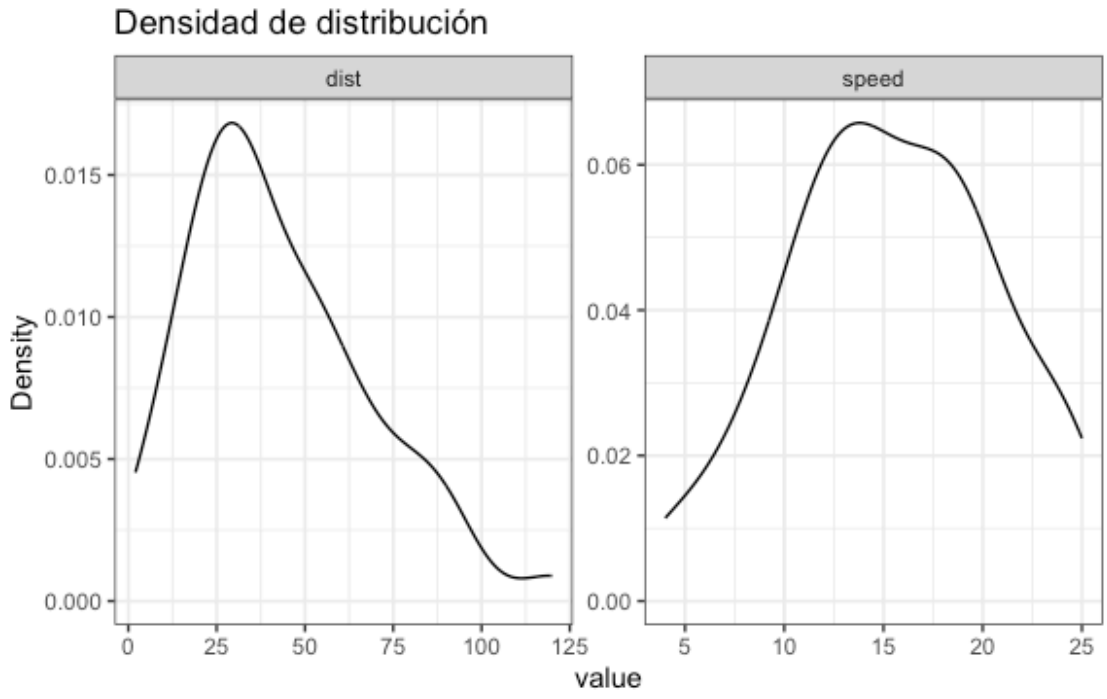
- *$speed* : velocidad medida en millas por hora

- *$dist* : distancia de frenado

```
#
#    los datos para el estudio de la regresión lineal simple los obtenemos del dataset cars,
# que muestra la distancia de frenado y velocidad (en la década de 1.920)
datos_rls <- cars
# descripción de las variables en el dataset
plot_intro (datos_rls,
            title = "Variables del dataset cars",
            ggtheme = theme_bw())
```




```
# distribución de densidad de $speed y $dist
plot_density(datos_rls,
             title = "Densidad de distribución",
             ggtheme = theme_bw())
```




```
# que estructura tienen nuestros datos, que tipo de datos cada variable
str(datos_rls)

'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...

# y comprobamos las primeras y últimas filas por si hay algún dato que nos llame la atención
head(datos_rls, 5)

  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16

#
tail(datos_rls, 5)
```

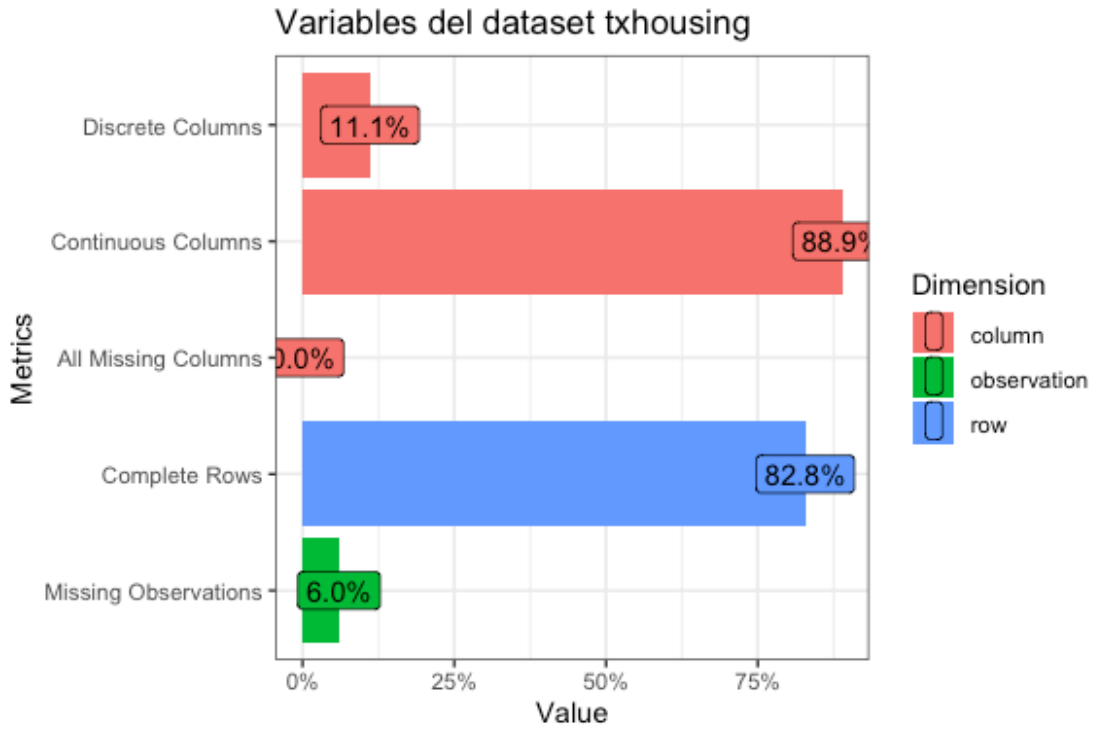
```
   speed dist
46    24   70
47    24   92
48    24   93
49    24  120
50    25   85
```

Regresión lineal polinómica.

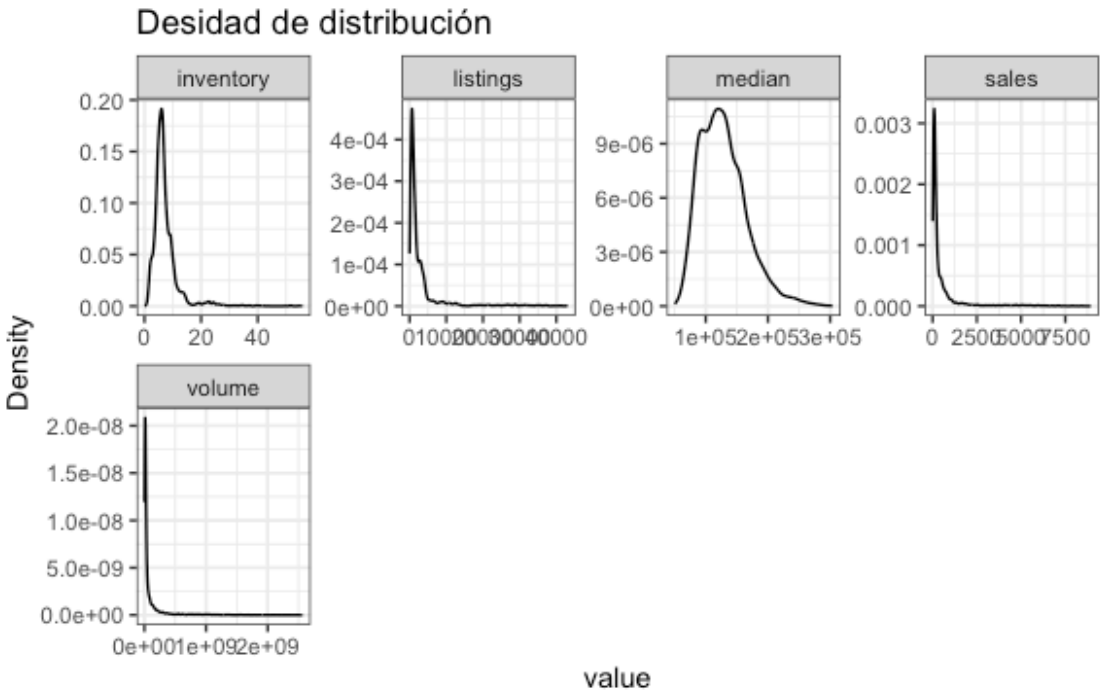
En este caso estudiamos la relación entre una variable independientes y varias variables dependientes de alguna forma relacionadas con la anterior, elegimos para esta publicación el dataset *txhousing* (disponible en el package *ggplot2* ((Wickham 2016a)), que contiene las siguientes variables :

- *$city* : nombre del área MLS (Multiple Listing Service)
- *$year, $month, $date* : fecha
- *$sales* : número de ventas
- *$volume* : valor total de las ventas
- *$median*: precio de venta mediano
- *$listings* : listados totales activos
- *$inventory* : Meses en stock tiempo que llevaría vender todas las casas disponibles al ritmo actual de ventas

```
#
#    los datos para la regresión lineal polinómica los obtendremos del dataset txhousing
#(disponible en el package ggplot2), que contiene 9 variables del mercado de la vivienda en
#Tejas.
datos_rlp <-txhousing
#
# descripción de las variables en el dataset
plot_intro (datos_rlp,
            title = "Variables del dataset txhousing",
            ggtheme = theme_bw())
```




```
# distribución de densidad de de las variables que contiene el dataset, limito el rango de
#columnas con información relevante (no necesito saber la densidad, del año, la fecha ...)
plot_density(datos_rlp[, 4:8],
             title = "Desidad de distribución",
             ggtheme = theme_bw())
```




```
# que estructura tienen nuestros datos, que tipo de datos cada variable
str(datos_rlp)

tibble [8,602 × 9] (S3: tbl_df/tbl/data.frame)
 $ city     : chr [1:8602] "Abilene" "Abilene" "Abilene" "Abilene" ...
 $ year     : int [1:8602] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
 $ month    : int [1:8602] 1 2 3 4 5 6 7 8 9 10 ...
 $ sales    : num [1:8602] 72 98 130 98 141 156 152 131 104 101 ...
 $ volume   : num [1:8602] 5380000 6505000 9285000 9730000 10590000 ...
 $ median   : num [1:8602] 71400 58700 58100 68600 67300 66900 73500 75000 64500 59300 ...
 $ listings : num [1:8602] 701 746 784 785 794 780 742 765 771 764 ...
 $ inventory: num [1:8602] 6.3 6.6 6.8 6.9 6.8 6.6 6.2 6.4 6.5 6.6 ...
 $ date     : num [1:8602] 2000 2000 2000 2000 2000 ...

# y comprobamos las primeras y últimas filas por si hay algún dato que nos llame la atención
head(datos_rlp, 5)

# A tibble: 5 × 9
  city     year month sales   volume median listings inventory  date
  <chr>   <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
1 Abilene  2000     1    72  5380000  71400      701       6.3 2000
```

```
2 Abilene  2000     2    98  6505000  58700      746       6.6 2000.
3 Abilene  2000     3   130  9285000  58100      784       6.8 2000.
4 Abilene  2000     4    98  9730000  68600      785       6.9 2000.
5 Abilene  2000     5   141 10590000  67300      794       6.8 2000.

#
tail(datos_rlp, 5)

# A tibble: 5 × 9
  city          year month sales   volume median listings inventory  date
  <chr>        <int> <int> <dbl>    <dbl>  <dbl>    <dbl>     <dbl> <dbl>
1 Wichita Falls  2015     3   152 16716584  89200      818       6.8 2015.
2 Wichita Falls  2015     4   129 15482194 105300      760       6.4 2015.
3 Wichita Falls  2015     5   174 19188181 100000      776       6.4 2015.
4 Wichita Falls  2015     6   143 18820752 118800      770       6.2 2015.
5 Wichita Falls  2015     7   172 23850905 116700      811       6.5 2016.
```

Como podemos ver en el gráfico obtenido con *plot_intro()*, tenemos un 6% de observaciones faltantes (en algunos textos figuran como N/A), en circunstancias reales deberíamos investigar el motivo de estas observaciones perdidas y tomar una decisión al respecto. Dado que el propósito es mostrar la metodología y desconocemos cómo se obtuvo el dataset txhousing, en este caso no prestaremos atención a estas variables faltantes.

Regresión lineal simple.

Este es el caso más sencillo de regresión lineal donde investigamos la relación entre dos variables con una ecuación del tipo :

$$y = a\,x + m + \epsilon$$

en esta ecuación, como ya conocemos :

- y es la variable independiente, en nuestro caso la distancia de frenado.
- x es la variable dependiente, en nuestro caso la velocidad del vehículo.
- a es la pendiente de la recta o también la variación de y por unidad de x.
- m es el punto de intersección con el eje de abcisas.
- ϵ es una variable independiente de media 0 y distribución tipo normal.

Como hemos visto anteriormente, mediante *ggplot2* ((Wickham 2016a)) podemos obtener gráficamente la solución a encontrar la relación lineal entre dos variables mediante la capa *geom_smooth()*, veamos como:

```
#
#    Presentamos el gráfico que relaciona ambas variables $speed y $dist, superponiendo la #recta entre ambas variables que calcula ggplot2
ggplot(data=datos_rls, aes(x=speed, y=dist))+
    geom_point(alpha=0.5)+   # alpha es el valor de transparencia en este caso de los puntos
    geom_smooth(method = lm, aes(color="Reg.Lin."))+
    scale_color_manual(values = c("Reg.Lin."="blue"))+
    labs(title = "Distancia de frenado vs. velocidad",
         subtitle = "Dataset: cars",
         caption = "Medida años 1.920")+
    xlab("Velocidad (mph)")+
    ylab("Distancia de frenado (ft)")
```

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

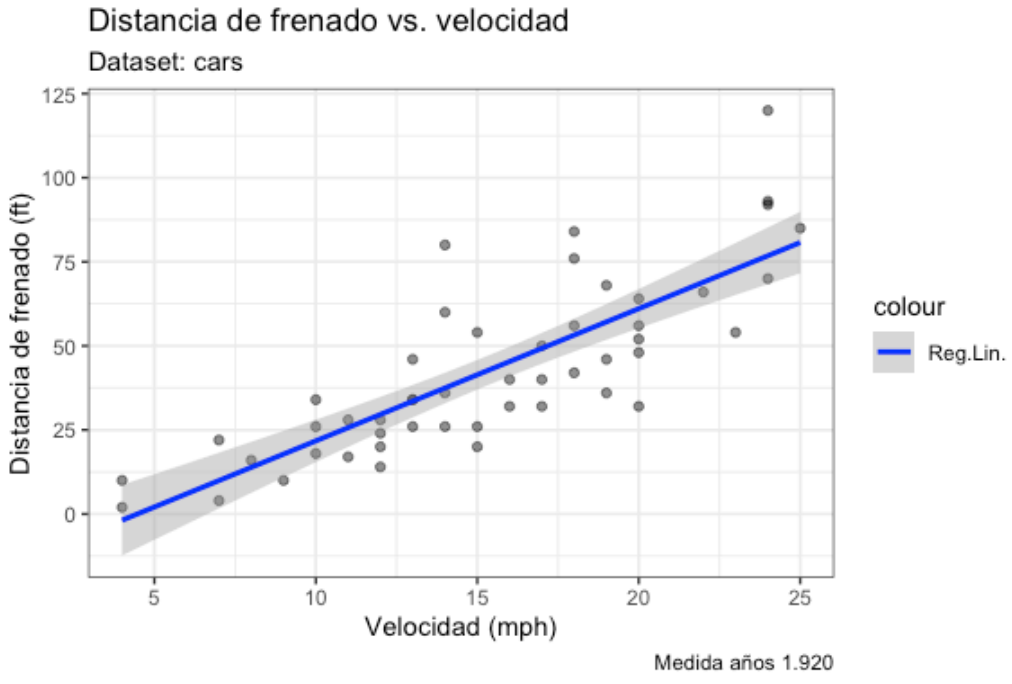


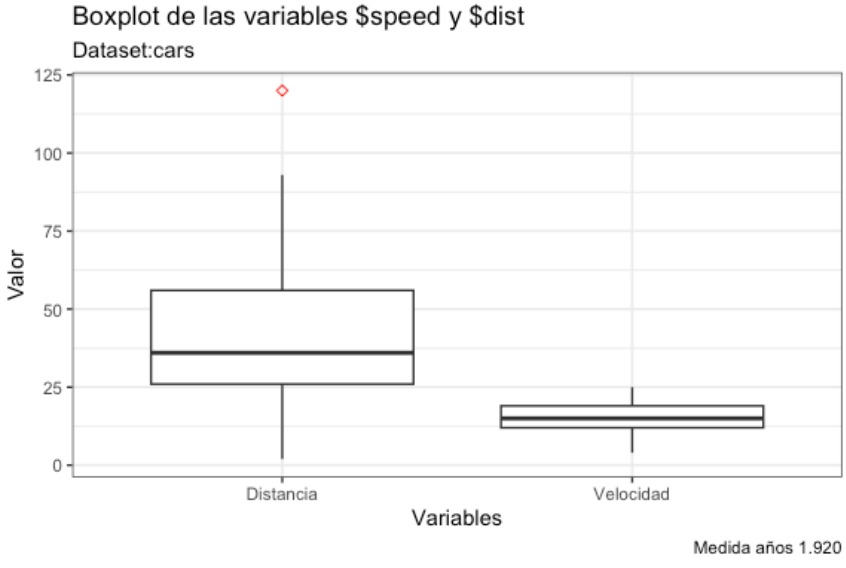
Figura 1: Relación distancia de frenado y velocidad.

En el gráfico Figura 1 vemos gráficamente representados los valores de ambas variables del dataframe *datos_rls*, y una recta en color azul que *ggplot2* ((Wickham 2016b)) calcula para relacionar ambas variables. La banda gris que rodea a la recta en azul es el error standard, que como ponemos ver por su anchura es mayor en los extremos que en el centro.

Gráficamente vemos que podemos obtener una recta que nos relacione ambas variables, que en el tramo central funcionará con un bajo error standard, y en los extremos el error será mayor porque los valores están bastante más alejados y no es posible capturarlos con una recta.

Como seguridad adicional, vamos a ver en gráfico de cajas (con la capa *geom_boxplot()*) ambas variables para ver si hay algún valor atípico (en ingles outlier)

```r
#
#       para mostrar los datos en el gráfico de caja vamos a añadir dos variables categoricas
#distancia y velocidad
cat_distancia <- rep ("Distancia", nrow(datos_rls))
cat_velocidad <- rep ("Velocidad", nrow(datos_rls))
# componemos dos dataframes con la categoria  de cada valor y los valores por categoria
df_distancia <- data.frame(categoria=cat_distancia, valor=datos_rls$dist)
df_velocidad <- data.frame(categoria=cat_velocidad, valor=datos_rls$speed)
# y los unimos a nivel fila con la función rbind() en un único dataframe que pasaremos al
# gráfico
datos_boxplot <- rbind(df_distancia, df_velocidad)

#
# pasamos los datos a geom_boxplot() usando un operador pipeline %>%
datos_boxplot %>%
    ggplot()+
    geom_boxplot(aes(x=categoria, y=valor),
                 outliers = TRUE, outlier.color = "red", outlier.shape = 5)+
    labs(title = "Boxplot de las variables $speed y $dist",
         subtitle = "Dataset:cars",
         caption="Medida años 1.920")+
    xlab("Variables")+
    ylab("Valor")
```



del gráfico Figura 2 en color rojo vemos identificado un valor atípico, para nuestra publicación no haremos nada con este valor atípico, en la vida real los valores outliers deben analizarse y tomarse decisiones sobre que hacer con ellos (forman parte del proceso habitual y deben seguir, es un error en la medida y debe repetirse, …). Es llamativo como en muchas ocasiones en la vida real, este tipo de valores simplemente no se analizan o no se les presta la atención debida.

En R ((2024a)) disponemos de la función *lm()*, para el cálculo de una regresión lineal, veamos a continuación como :

```
#
#      mediante la función lm(), hacemos una recta donde $dist es función lineal de $speed, del
#dataframe datos_rls
modelo_rls <- lm (dist ~ speed, data=datos_rls)
# vemos con la función summary() los valores del modelo que obtenemos de aplicar lm()
summary(modelo_rls)


Call:
lm(formula = dist ~ speed, data = datos_rls)

Residuals:
    Min      1Q  Median      3Q     Max
-29.069  -9.525  -2.272   9.215  43.201

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

Del resumen obtenido con la función *summary()* tenemos las siguientes secciones:

- Call : muestra como se ha sido la llamada de la función *lm()*

- Residuals: los errores del modelo, si los valores de error fueran adecuados tal como hemos definido teóricamente la mediana (en una distribución normal es igual a la media) debería ser cero, en este caso vemos que no es así.
- coeficientes: los coeficientes de nuestra recta, junto con su p-value
- la ultima sección son las medidas de ajuste del modelo obtenido

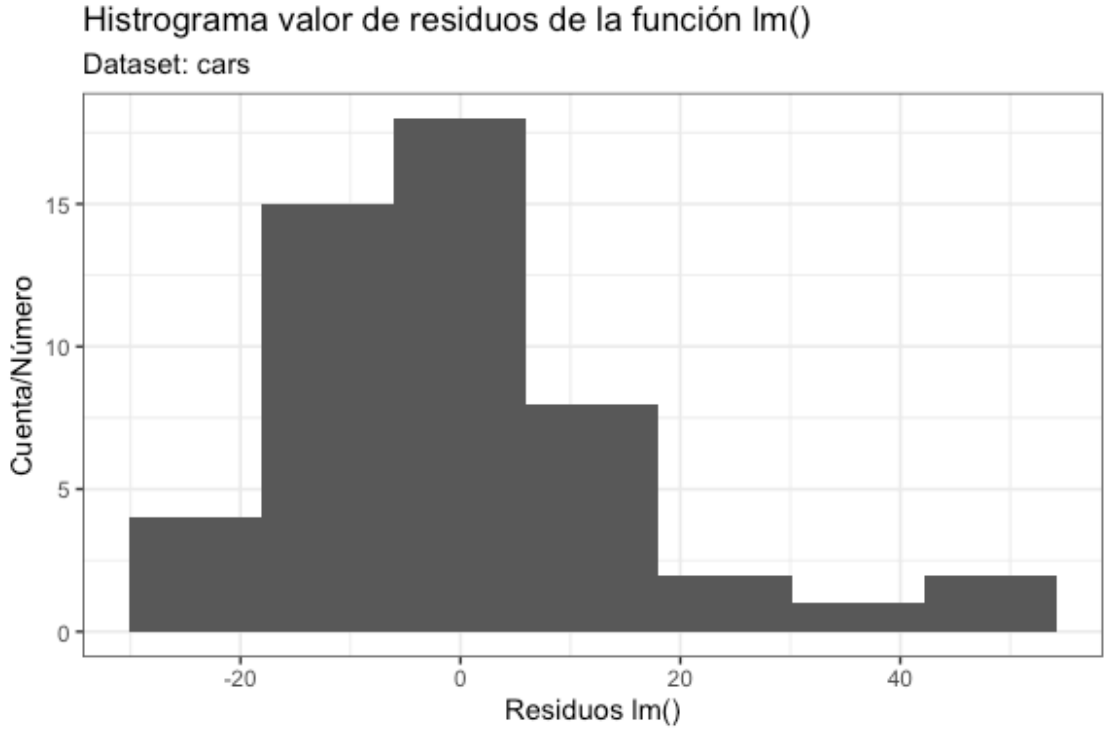
R ((2024a)) nos permite obtener del modelo información adicional que muestra el grado de ajuste que hemos logrado.

```
#    R nos permite obtener ciertas partes del modelo que hemos creado
# los coeficientes de la recta
coef_rls <- coefficients(modelo_rls)
coef_rls

(Intercept)       speed
 -17.579095    3.932409

# los valores de la variable independiente que calcula el modelo
head(fitted(modelo_rls), 5)

       1        2        3        4        5
-1.849460 -1.849460  9.947766  9.947766 13.880175

# los valores residuos, la diferencia entre la variable independiente real y la calculada
head(resid(modelo_rls), 5)

       1        2        3        4        5
 3.849460 11.849460 -5.947766 12.052234  2.119825
```

Si hacemos un histograma de los residuos, vemos que la mayor parte de ellos tienen el valor 0 (el modelo predice la variable real)

```
#
#    paso a un data frame los datos de los residuos
residuos <- as.data.frame(resid(modelo_rls))
#    a traves del operador %>% paso los datos a un geom_histogram de ggplot
residuos %>%
    ggplot()+
    geom_histogram(aes(x=resid(modelo_rls)), bins= round(sqrt(nrow(residuos)),0))+
    labs(title = "Histrograma valor de residuos de la función lm()",
         subtitle= "Dataset: cars")+
```

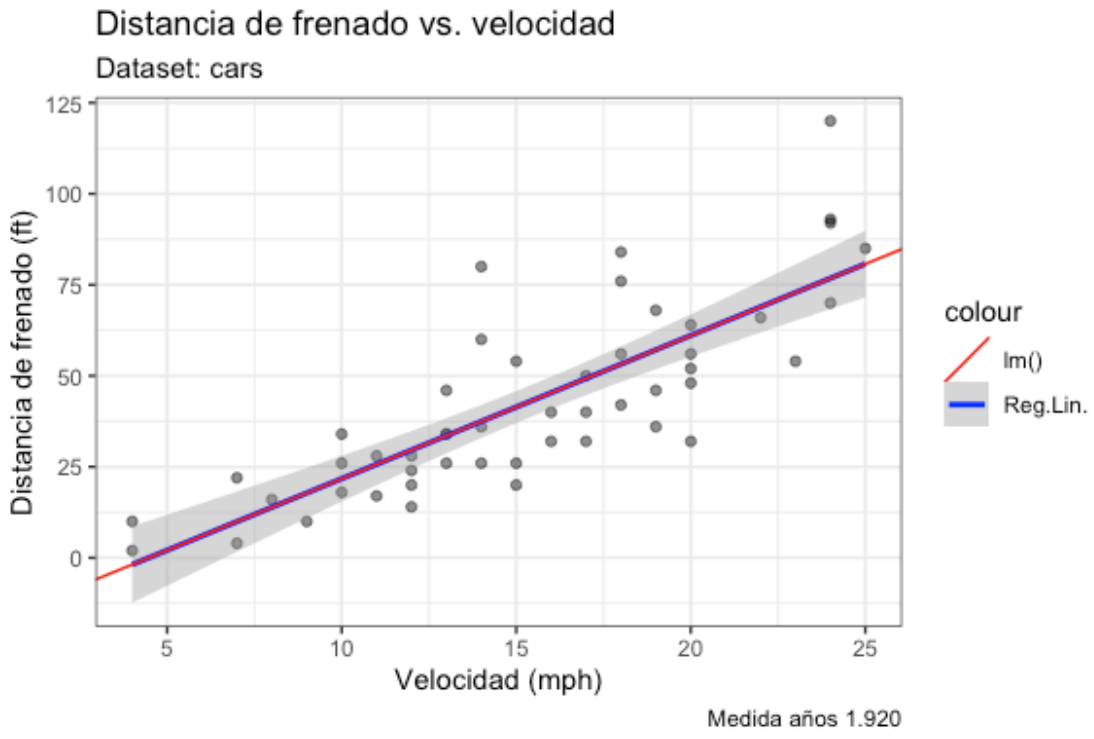
```
  xlab("Residuos lm()")+
  ylab("Cuenta/Número")
```



Vamos a ver gráficamente el resultado del modelo aprovechando las capacidades de *ggplot2* ((Wickham 2016c)) , superponiéndolo con *geom_abline()* sobre Figura 1:

```
#
# sobre el gráfico anterior añadimos una capa adicional mostrando la recta del modelo calculado
#con lm()
datos_rls %>%
    ggplot(aes(x=speed, y=dist))+
        geom_point(alpha=0.5)+   # alpha es el valor de transparencia en este caso de los puntos
        geom_smooth(method = lm, aes(color="Reg.Lin."))+
        geom_abline(aes(intercept = coef_rls[1], slope=coef_rls[2], color="lm()"))+
        scale_color_manual(values = c("Reg.Lin."="blue", "lm()"="red"))+
        labs(title = "Distancia de frenado vs. velocidad",
            subtitle = "Dataset: cars",
            caption = "Medida años 1.920")+
        xlab("Velocidad (mph)")+
        ylab("Distancia de frenado (ft)")
```

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



Regresión lineal polinómica.

En este caso, como hemos indicado anteriormente usaremos el data set *txhousing*, que nos da varias variables del mercado inmobiliario de Texas. Vamos a trabajar para relacionar más de dos variables, aprovecharemos para presentar algunas funciones del package ggally ((Schloerke et al. 2024)).

```
#
#    en este tipo de casos que trabajamos con muchas variables, recurrimos a la función
# ggpairs() del package ggally, para ver gráficamente algunas relaciones entre variables.
ggpairs(datos_rlp[, 4:8])
```

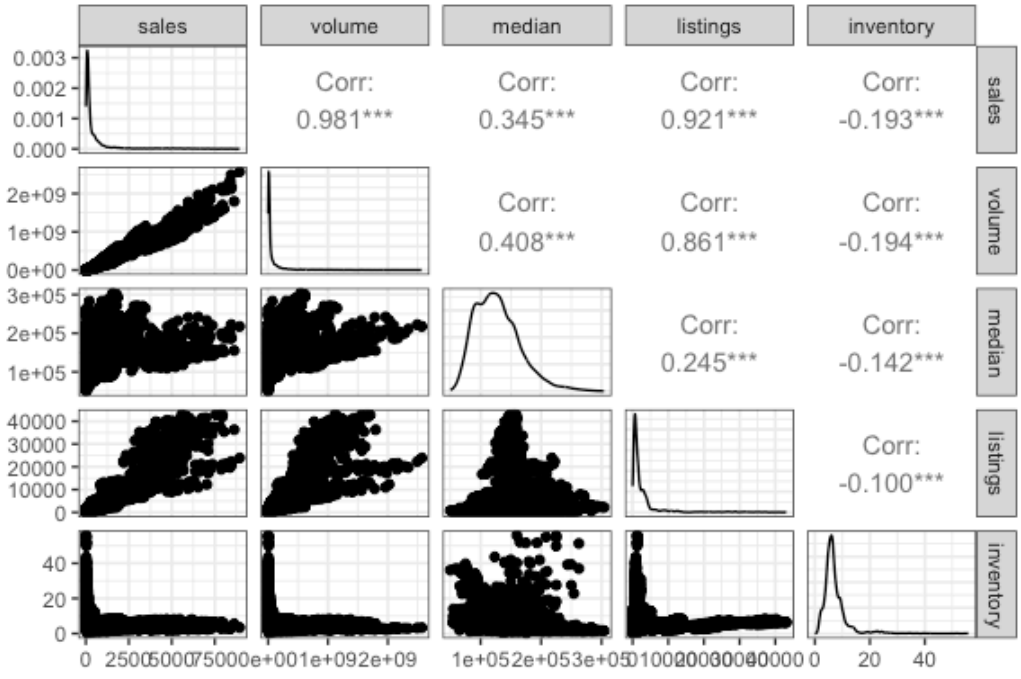


Figura 3: Relación entre variables del dataset txhousing

Del gráfico, podemos ver que el número de ventas (*$sales*) correla en distinto grado (recordemos que el coeficiente de correlación como máximo es abs(r)=1) con el valor total de las ventas (*$volume*) , con el precio mediano de venta (*$median*) y con el stock activo de viviendas (*$listings*), vamos a ver como hacer un modelo lineal entre estas cuatro variables.

```
#
#    almacenamos en una variable el resultado de la función lm() donde mediante el operador #tilde
(~) definimos la relación que queremos modelar
modelo_rlp <- lm(sales ~ volume+median+listings, data=datos_rlp)
# obtenemos la información del modelo con summary()
summary (modelo_rlp)
```

```
Call:
lm(formula = sales ~ volume + median + listings, data = datos_rlp)

Residuals:
    Min      1Q  Median      3Q     Max 
-1172.86   -45.05   -11.99    33.58  1553.30 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1.699e+02  6.308e+00   26.93   <2e-16 ***
volume       3.402e-06  1.345e-08  252.95   <2e-16 ***
median      -1.130e-03  4.791e-05  -23.58   <2e-16 ***
listings     5.516e-02  5.462e-04  101.00   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 137.2 on 7164 degrees of freedom
  (1434 observations deleted due to missingness)
Multiple R-squared:  0.9861,    Adjusted R-squared:  0.986 
F-statistic: 1.688e+05 on 3 and 7164 DF,  p-value: < 2.2e-16
```

como vemos, tenemos las mismas secciones que en el modelo con dos variables.

- Call : muestra como se ha sido la llamada de la función *lm()*
- Residuals: los errores del modelo, los valores de error fueran adecuados tal como hemos definido teóricamente la mediana (en una distribución normal es igual a la media) debería ser cero, en este caso vemos que no es así.
- Coeficientes: los coeficientes de nuestra recta, junto con su p-value
- la ultima sección son las medidas de ajuste del modelo obtenido. Aquí merece la pena destacar el elevado valor de R-squared que mide el porcentaje de variación de *$sales*, que nuestro modelo es capaz de detectar, con un 98% estamos hablando de un modelo bastante ajustado.

Tal y como hicimos con el caso de regresión lineal simple, R ((2024b)) nos proporciona información adicional sobre el modelo que hemos calculado con la función *lm()*

```
# R nos permite obtener ciertas partes del modelo que hemos creado
# los coeficientes de la recta
```

```
coef_rlp <- coefficients(modelo_rlp)
coef_rlp
```

```
 (Intercept)       volume       median     listings
1.698853e+02 3.402285e-06 -1.129921e-03 5.516402e-02
```

```
# los valores de la variable independiente que calcula el modelo (lo limitamos a 5, el data set tiene 8.602 observaciones )
print ("Valor variable indep. calculada")
```

```
[1] "Valor variable indep. calculada"
```

```
head(fitted(modelo_rlp), 5)
```

```
      1       2       3       4       5
146.1832 166.8431 179.0757 168.7807 173.6720
```

```
tail(fitted(modelo_rlp), 5)
```

```
   8598    8599    8600    8601    8602
171.0951 145.5041 164.9841 142.1605 163.9091
```

```
# los valores residuos, la diferencia entre la variable independiente real y la calculada
print ("Valor residuos calculados")
```

```
[1] "Valor residuos calculados"
```

```
head(resid(modelo_rlp), 5)
```

```
       1        2        3        4        5
-74.18318 -68.84313 -49.07567 -70.78068 -32.67202
```

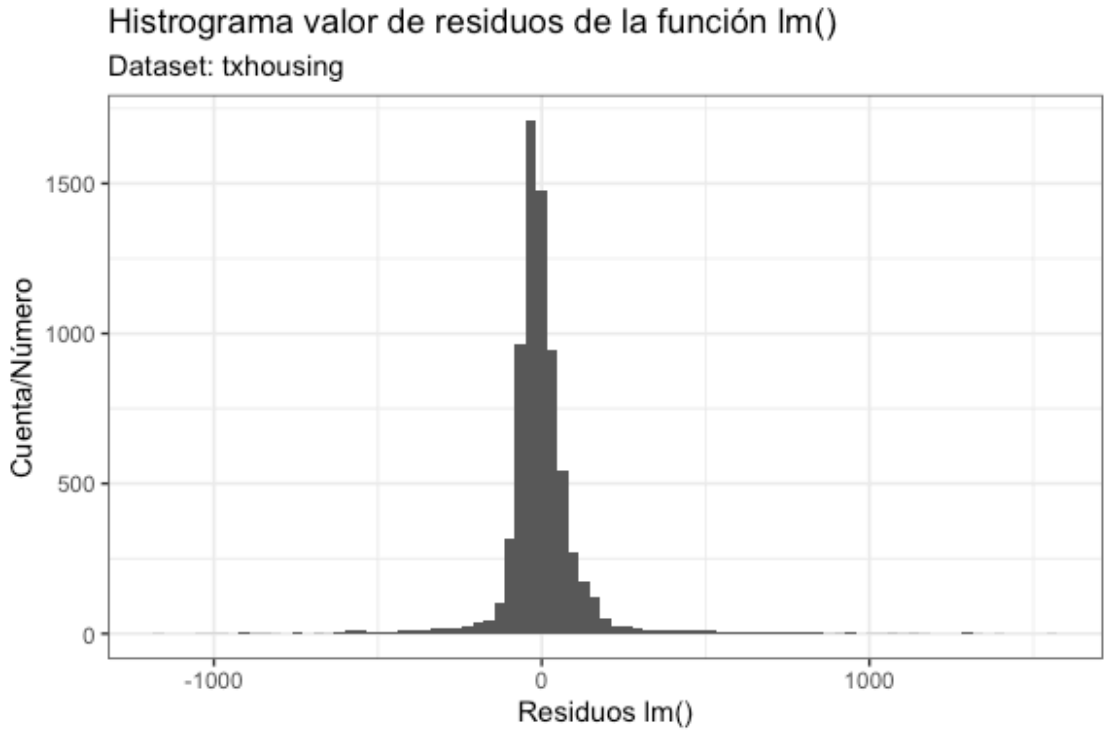
```
tail(resid(modelo_rlp), 5)
```

```
      8598       8599       8600       8601       8602
-19.0950775 -16.5040899   9.0158790   0.8394753   8.0908999
```

Hacemos como el caso anterior y vemos mediante un histograma como se distribuye el valor de los residuos de la ecuación calculada :

```
#
#    paso a un data frame los datos de los residuos
residuos <- as.data.frame(resid(modelo_rlp))
#    a traves del operador %>% paso los datos a un geom_histogram de ggplot
residuos %>%
    ggplot()+
        geom_histogram( aes(x=resid(modelo_rlp)),
```

```
                bins= round(sqrt(nrow(residuos)),0))+
  labs(title = "Histrograma valor de residuos de la función lm()",
       subtitle= "Dataset: txhousing")+
  xlab("Residuos lm()")+
  ylab("Cuenta/Número")
```



Vemos que el modelo, tal y como esperábamos del valor R-squared obtenido, obtiene en el cero la mayoría del valor de residuos.

Algunos puntos adicionales…

- La regresión lineal asume que la variable independiente es el resultado de una combinación lineal de las variables dependientes. Esta asunción funciona cuando es “casi” verdad, pero en algunas ocasiones puede funcionar bien sin serlo.
- Si se quieren usar los coeficientes de la ecuación obtenidos para hacer una extrapolación, deben ser estadísticamente significativos.
- Dependiendo de la aplicación este tipo de modelos deben usarse con cuidado en el caso de planear con ellos un a extrapolación. Aún en el caso de que en el rango de

valores donde hemos calculado el modelo el ajuste sea bueno, eso no asegura que siga siendo bueno más allá de ese rango de valores.

- Valores muy grandes de coeficientes o de p-value, pueden ser indicativos de autocorrelación entre variables dependientes.
- Hemos usado en esta publicación la forma más simple de realizar la regresión lineal, normalmente el conjunto de datos disponible hay que dividirlo en un conjunto de entrenamiento, y otro de validación, entrenar el modelo, ….

Referencias

R Core Team. 2024a. «R: A Language and Environment for Statistical Computing». https://www.R-project.org/.

———. 2024b. «R: A Language and Environment for Statistical Computing». https://www.R-project.org/.

Schloerke, Barret, Di Cook, Joseph Larmarange, Francois Briatte, Moritz Marbach, Edwin Thoen, Amos Elberg, y Jason Crowley. 2024. «GGally: Extension to 'ggplot2'». https://CRAN.R-project.org/package=GGally.

Wickham, Hadley. 2016a. «ggplot2: Elegant Graphics for Data Analysis». https://ggplot2.tidyverse.org.

———. 2016b. «ggplot2: Elegant Graphics for Data Analysis». https://ggplot2.tidyverse.org.

———. 2016c. «ggplot2: Elegant Graphics for Data Analysis». https://ggplot2.tidyverse.org.