Modelos Lineales en R

Datos de Casas en Ames, Iowa US

Estos datos contienen 2930 casas en la ciudad de Ames en Iowa en Estados Unidos. Los datos originales han sido modificados para facilitar uso y hacen parte del paquete modeldata. *modeldata* hace parte de tidyverse

Usaremos tambien un nuevo paquete mapview. Este paquete nos servira para visualizar datos Geospaciales muy facilmente.

Los siguientes comandos son necesarios en linux como prerequisito para instalar el paquete mapview

```
sudo apt-get install libudunits2-dev
sudo apt-add-repository ppa:ubuntugis/ubuntugis-unstable
sudo apt-get update
sudo apt-get install libgdal-dev libgeos-dev libproj-dev
```

Importemos las librerias necesarias

```
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.2.0 --
```

```
## v broom        0.7.12     v recipes      0.2.0
## v dials        0.1.0      v rsample      0.1.1
## v dplyr        1.0.8      v tibble       3.1.6
## v ggplot2      3.3.5      v tidyr        1.2.0
## v infer        1.0.0      v tune         0.2.0
## v modeldata    0.1.1      v workflows    0.2.6
## v parsnip      0.2.1      v workflowsets 0.2.1
## v purrr        0.3.4      v yardstick    0.0.9
```

```
## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```
library(mapview)
```

Resolvamos los conflictos entre los distintos paquetes

```
tidymodels_prefer(quiet=FALSE)
```

```
## [conflicted] Will prefer dplyr::filter over any other package
## [conflicted] Will prefer dplyr::select over any other package
## [conflicted] Will prefer dplyr::slice over any other package
## [conflicted] Will prefer dplyr::rename over any other package
## [conflicted] Will prefer dials::neighbors over any other package
## [conflicted] Will prefer parsnip::fit over any other package
## [conflicted] Will prefer parsnip::bart over any other package
## [conflicted] Will prefer parsnip::pls over any other package
```

```
## [conflicted] Will prefer purrr::map over any other package
## [conflicted] Will prefer recipes::step over any other package
## [conflicted] Will prefer themis::step_downsample over any other package
## [conflicted] Will prefer themis::step_upsample over any other package
## [conflicted] Will prefer tune::tune over any other package
## [conflicted] Will prefer yardstick::precision over any other package
## [conflicted] Will prefer yardstick::recall over any other package
## [conflicted] Will prefer yardstick::spec over any other package
## -- Conflicts ------------------------------------------ tidymodels_prefer() --
```

Carguemos los datos

```
data(ames)

head(ames)
```

```
## # A tibble: 6 x 74
##   MS_SubClass             MS_Zoning Lot_Frontage Lot_Area Street Alley Lot_Shape
##   <fct>                   <fct>            <dbl>    <int> <fct>  <fct> <fct>
## 1 One_Story_1946_and_New~ Resident~          141    31770 Pave   No_A~ Slightly~
## 2 One_Story_1946_and_New~ Resident~           80    11622 Pave   No_A~ Regular
## 3 One_Story_1946_and_New~ Resident~           81    14267 Pave   No_A~ Slightly~
## 4 One_Story_1946_and_New~ Resident~           93    11160 Pave   No_A~ Regular
## 5 Two_Story_1946_and_New~ Resident~           74    13830 Pave   No_A~ Slightly~
## 6 Two_Story_1946_and_New~ Resident~           78     9978 Pave   No_A~ Slightly~
## # ... with 67 more variables: Land_Contour <fct>, Utilities <fct>,
## #   Lot_Config <fct>, Land_Slope <fct>, Neighborhood <fct>, Condition_1 <fct>,
## #   Condition_2 <fct>, Bldg_Type <fct>, House_Style <fct>, Overall_Cond <fct>,
## #   Year_Built <int>, Year_Remod_Add <int>, Roof_Style <fct>, Roof_Matl <fct>,
## #   Exterior_1st <fct>, Exterior_2nd <fct>, Mas_Vnr_Type <fct>,
## #   Mas_Vnr_Area <dbl>, Exter_Cond <fct>, Foundation <fct>, Bsmt_Cond <fct>,
## #   Bsmt_Exposure <fct>, BsmtFin_Type_1 <fct>, BsmtFin_SF_1 <dbl>, ...
```

Revisemos las dimensiones del dataframe

```
dim(ames)
```

```
## [1] 2930   74
```

Revisemos la lista de variables

```
names(ames)
```

```
##  [1] "MS_SubClass"        "MS_Zoning"          "Lot_Frontage"
##  [4] "Lot_Area"           "Street"             "Alley"
##  [7] "Lot_Shape"          "Land_Contour"       "Utilities"
## [10] "Lot_Config"         "Land_Slope"         "Neighborhood"
## [13] "Condition_1"        "Condition_2"        "Bldg_Type"
## [16] "House_Style"        "Overall_Cond"       "Year_Built"
## [19] "Year_Remod_Add"     "Roof_Style"         "Roof_Matl"
## [22] "Exterior_1st"       "Exterior_2nd"       "Mas_Vnr_Type"
## [25] "Mas_Vnr_Area"       "Exter_Cond"         "Foundation"
## [28] "Bsmt_Cond"          "Bsmt_Exposure"      "BsmtFin_Type_1"
## [31] "BsmtFin_SF_1"       "BsmtFin_Type_2"     "BsmtFin_SF_2"
## [34] "Bsmt_Unf_SF"        "Total_Bsmt_SF"      "Heating"
## [37] "Heating_QC"         "Central_Air"        "Electrical"
## [40] "First_Flr_SF"       "Second_Flr_SF"      "Gr_Liv_Area"
## [43] "Bsmt_Full_Bath"     "Bsmt_Half_Bath"     "Full_Bath"
```

```
## [46] "Half_Bath"           "Bedroom_AbvGr"      "Kitchen_AbvGr"
## [49] "TotRms_AbvGrd"       "Functional"         "Fireplaces"
## [52] "Garage_Type"         "Garage_Finish"      "Garage_Cars"
## [55] "Garage_Area"         "Garage_Cond"        "Paved_Drive"
## [58] "Wood_Deck_SF"        "Open_Porch_SF"      "Enclosed_Porch"
## [61] "Three_season_porch" "Screen_Porch"       "Pool_Area"
## [64] "Pool_QC"             "Fence"              "Misc_Feature"
## [67] "Misc_Val"            "Mo_Sold"            "Year_Sold"
## [70] "Sale_Type"           "Sale_Condition"     "Sale_Price"
## [73] "Longitude"           "Latitude"
```

```
class(ames$Bsmt_Full_Bath)
```

```
## [1] "numeric"
```

```
summary(ames$Bsmt_Full_Bath)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4311  1.0000  3.0000
```

Esto muestra la importancia de un diccionario de datos.

- Bsmt_Full_Bath: El sotano tiene un banno completo. Tipo: numerica. min = 0. max = 3.
- Firepalces: Chimeneas Tipo Integer min 0 max 4

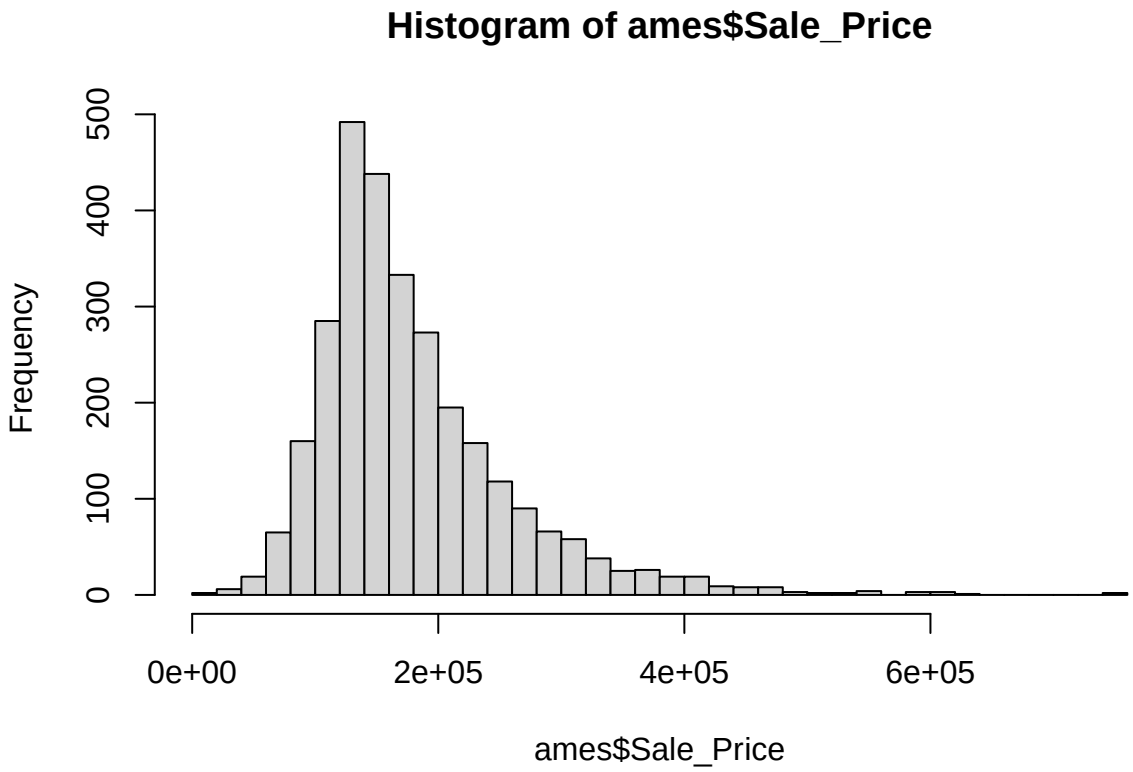
Pregunta Problema

Que queremos modelar?

Queremos estimar el precio de venta de las casas usando las otras variables como entradas del modelo

Revisemos el histograma de la variable que queremos estimar

```
hist(ames$Sale_Price, breaks = 50)
```




```
summary(ames$Sale_Price)
```

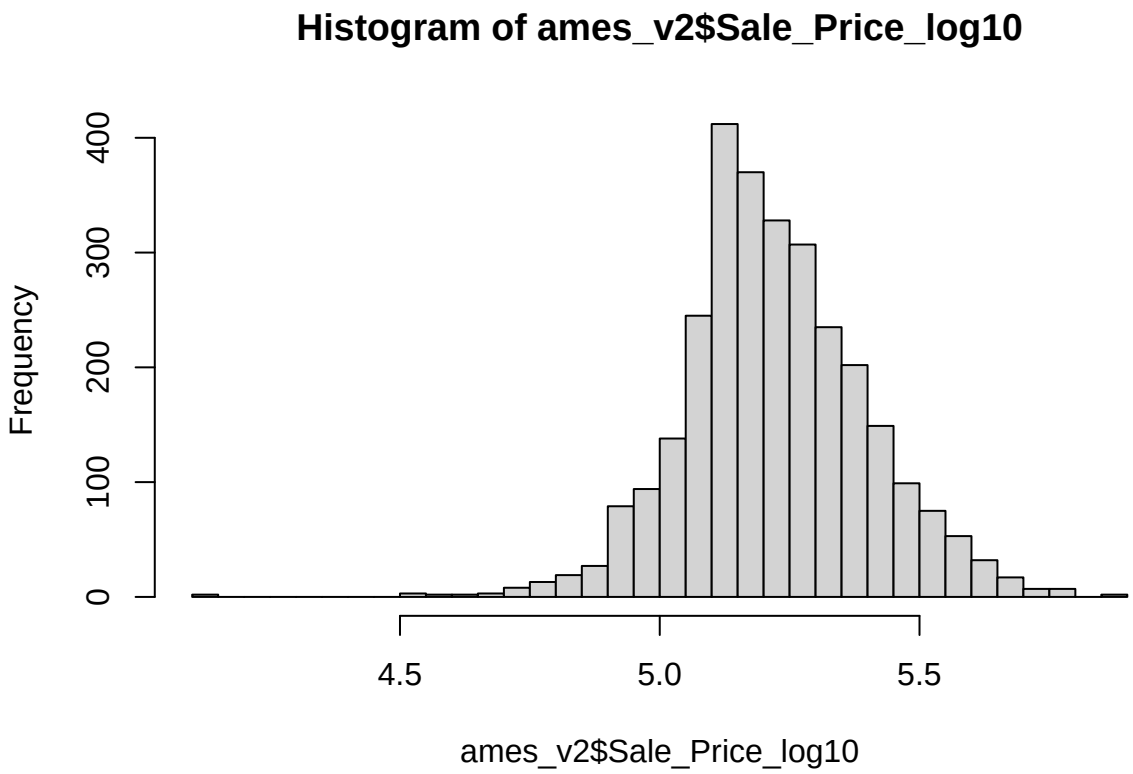
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12789  129500  160000  180796  213500  755000
```

Dado que la distribucion esta significativamente *skewed* vamos a utilizar una transformacion logaritmica en la salida. Para esto creemos una nueva variable con dicha transformacion

```
ames_v2 <- ames %>%
  mutate(Sale_Price_log10 = log10(Sale_Price))
```

Y ahora grafiquemos el Histograma

```
hist(ames_v2$Sale_Price_log10, breaks = 50)
```




```
summary(ames_v2$Sale_Price_log10)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.107   5.112   5.204   5.221   5.329   5.878
```

Grafiquemos en un Mapa usando Latitud y Longitud

Miremos las latitudes y longitudes

```
ames_v2 %>%
  select(Latitude, Longitude, Neighborhood) %>%
  head(10)
```

```
## # A tibble: 10 x 3
##    Latitude Longitude Neighborhood
##       <dbl>     <dbl> <fct>
##  1     42.1     -93.6 North_Ames
##  2     42.1     -93.6 North_Ames
##  3     42.1     -93.6 North_Ames
##  4     42.1     -93.6 North_Ames
##  5     42.1     -93.6 Gilbert
##  6     42.1     -93.6 Gilbert
##  7     42.1     -93.6 Stone_Brook
##  8     42.1     -93.6 Stone_Brook
##  9     42.1     -93.6 Stone_Brook
## 10     42.1     -93.6 Gilbert
```

Podemos copiar y pegar la latitud y la longitud de una de las casas en Google Maps y observar que si son datos reales

- Entremos a Google Maps
- Copiemos el siguiente texto `42.05403, -93.61975`

Primero seleccionemos un subconjunto de los barrios. Para esto miremos que barrios hay

```
levels(ames_v2$Neighborhood)
```

```
##  [1] "North_Ames"
##  [2] "College_Creek"
##  [3] "Old_Town"
##  [4] "Edwards"
##  [5] "Somerset"
##  [6] "Northridge_Heights"
##  [7] "Gilbert"
##  [8] "Sawyer"
##  [9] "Northwest_Ames"
## [10] "Sawyer_West"
## [11] "Mitchell"
## [12] "Brookside"
## [13] "Crawford"
## [14] "Iowa_DOT_and_Rail_Road"
## [15] "Timberland"
## [16] "Northridge"
## [17] "Stone_Brook"
## [18] "South_and_West_of_Iowa_State_University"
## [19] "Clear_Creek"
## [20] "Meadow_Village"
## [21] "Briardale"
## [22] "Bloomington_Heights"
## [23] "Veenker"
## [24] "Northpark_Villa"
## [25] "Blueste"
## [26] "Greens"
## [27] "Green_Hills"
## [28] "Landmark"
## [29] "Hayden_Lake"
```

Ahora creemos el subconjunto

```
ames_v2_subset <- ames_v2 %>%
  filter(Neighborhood %in% c("North_Ames", "Old_Town", "Somerset"))
```

Ahora si graficamos el mapa

```
mapview(ames_v2_subset,
        xcol = "Longitude",
        ycol = "Latitude",
        zcol = "Neighborhood",
        legend = FALSE,
        crs = 4269,
        grid = FALSE,
        color = heat.colors(n = 3, alpha = 1),
        cex = 2.0,
        burst = TRUE)@map
```

```
## QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-rstudio-user'
## TypeError: Attempting to change the setter of an unconfigurable property.
## TypeError: Attempting to change the setter of an unconfigurable property.
```

Aqui una guia de cada parametro

- xcol: variable que se usa para el eje x.
- ycol: variable que se usa para el eje y.
- zcol: variable categorica que se usa para diferenciar los puntos en el mapa
- legend: opcion binaria (TRUE/FALSE) para mostrar o no una leyenda
- crs: Coordinate Reference System
- grid: opcion binaria (TRUE/FALSE) para mostrar la grilla en el dibujo
- color: paleta de colores para graficar los puntos. En este caso se usa una funcion base de R que contiene una paleta de colores que hace parte del paquete grDevices. Aqui una guia para el uso de paletas de colores en R.
- cex: controla el tamano de los puntos.
- burst: muestra todas las capas

Como utilizar los datos para ajustar el modelo

Usualmente es necesario dividir la informacion disponible en al menos dos conjuntos: *entrenamiento y prueba*. Con el primer conjunto tenemos la oportunidad de ajustar los parametros del modelo. Con el segundo conjunto tenemos la posibilidad de estimar que tan bueno es el modelo cuando lo usemos en la practica.

Particion Aleatoria

Afortundamente tidymodels tiene una funcion que nos facilita esto. Si no hay ninguna consideracion respecto a tiempo usualmente se hace una particion aleatoria de los datos disponibles. Dado que la particion es aleatoria usualmente hacemos algo que se llama *fijar la semilla* del generador de numeros aleatorios de R. Esto lo hacemos a traves de la funcion `set.seed()`. Esto nos permite reproducir los resultados nuevamente dado que se escogeran las mismas muestras para entrenamiento y prueba. Tradicionalmente se usa el 80%

para entrenamiento y el 20% para prueba. la funcion que usaremos hace parte del paquete rsample

```
set.seed(123)
ames_split <- initial_split(ames_v2, prop = 0.80)
ames_split
```

```
## <Analysis/Assess/Total>
## <2344/586/2930>
```

`ames_split` solo contiene informacion respecto a la particion. Si queremos tener realmente la particion de los datos necesitamos hacer lo siguiente

```
2344/2930
```

```
## [1] 0.8
```

```
ames_train <- training(ames_split)
ames_test  <- testing(ames_split)
dim(ames_train)
```

```
## [1] 2344   75
```

```
dim(ames_test)
```

```
## [1] 586  75
```

```
586/2930
```

```
## [1] 0.2
```

Particion Estratificada

No obstante, muchas veces es mejor usar una particion estratificada. Esta consiste en dividir a la variable continua en quartiles y garantizar que la proporcion de muestras en cada cuartil sea la misma para el conjunto de datos de entrenamiento y de prueba.

Primero calculemos los cuartiles

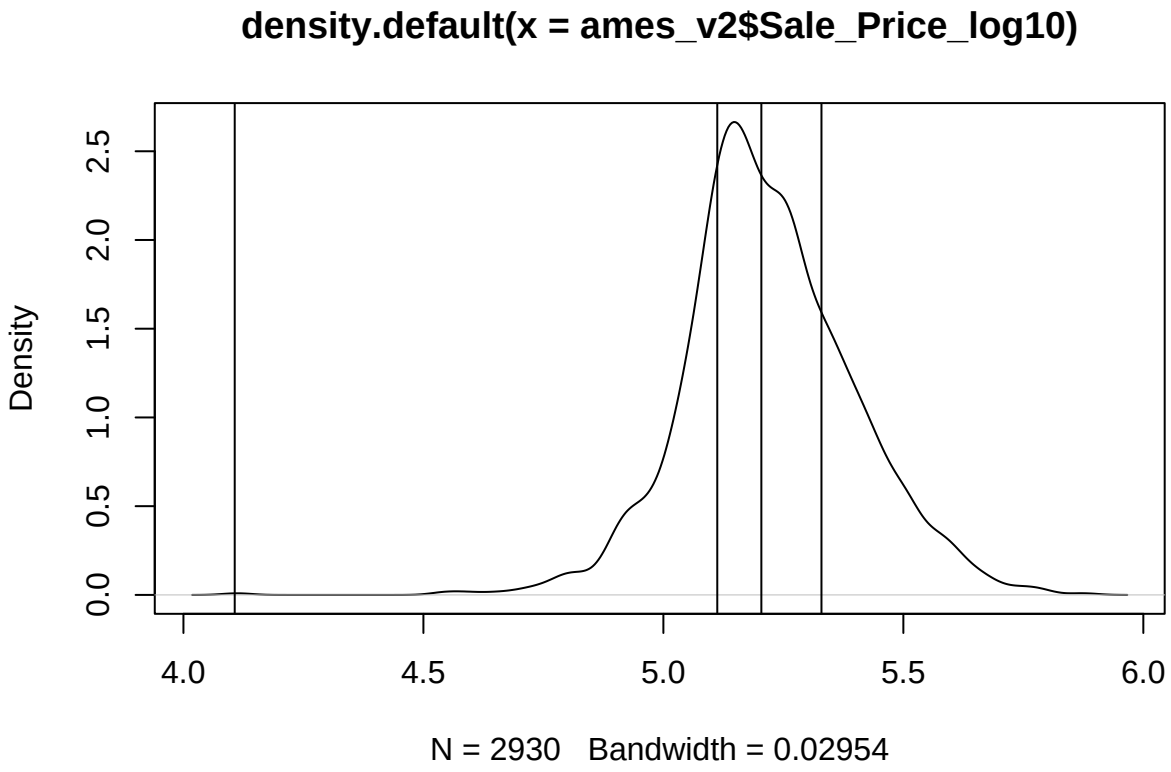
```
sales_quantiles <- quantile(ames_v2$Sale_Price_log10)

sales_quantiles
```

```
##       0%      25%      50%      75%     100%
## 4.106837 5.112270 5.204120 5.329398 5.877947
```

Ahora hagamos una grafica de densidad en vez de un histograma y visualizemos los cuartiles con lineas

```
plot(density(ames_v2$Sale_Price_log10))
abline(v=sales_quantiles[1])
abline(v=sales_quantiles[2])
abline(v=sales_quantiles[3])
abline(v=sales_quantiles[4])
```




```
set.seed(123)
ames_split <- initial_split(ames_v2, prop = 0.80, strata = Sale_Price_log10)
ames_train <- training(ames_split)
ames_test  <- testing(ames_split)
```

Verifiquemos los cuartiles de cada uno de los conjuntos

```
quantile(ames_train$Sale_Price_log10)
```

```
##       0%      25%      50%      75%     100%
## 4.106837 5.112270 5.204120 5.329398 5.872156
```

```
quantile(ames_test$Sale_Price_log10)
```

```
##       0%      25%      50%      75%     100%
## 4.544068 5.112270 5.204798 5.329091 5.877947
```

Particion usando tiempo

En algunas ocasiones el tiempo hay que considerarlo debido a la naturaleza de como se genera la informacion. Miremos la distribucion de cuando se vendieron las casas

```
table(ames_v2$Year_Sold)
```

```
##
## 2006 2007 2008 2009 2010
##  625  694  622  648  341
```

Ahora ordemos por Year. Esto es importante para que la funcion opere apropiadamente

```
ames_v3 <- ames_v2 %>%
  arrange(Year_Sold, Mo_Sold)
```

```
ames_v3 %>%
  select(Year_Sold, Mo_Sold) %>%
    head(20)
```

```
## # A tibble: 20 x 2
##    Year_Sold Mo_Sold
##        <int>   <int>
##  1      2006       1
##  2      2006       1
##  3      2006       1
##  4      2006       1
##  5      2006       1
##  6      2006       1
##  7      2006       1
##  8      2006       1
##  9      2006       1
## 10      2006       1
## 11      2006       1
## 12      2006       1
## 13      2006       1
## 14      2006       1
## 15      2006       1
## 16      2006       1
## 17      2006       1
## 18      2006       1
## 19      2006       2
## 20      2006       2
```

```
set.seed(123)
ames_split <- initial_time_split(ames_v3, lag=2)
ames_train <- training(ames_split)
ames_test  <- testing(ames_split)
```

Verifiquemos el Year min y max para cada set

```
c(min(ames_train$Year_Sold), max(ames_train$Year_Sold))
```

```
## [1] 2006 2009
```

```
c(min(ames_test$Year_Sold), max(ames_test$Year_Sold))
```

```
## [1] 2009 2010
```

Como crear un modelo