# Kaggle House Prices

*Javier Guzmán Figueira Domínguez*

*04/03/2018*

## Introducción

El objetivo de este documento es el realizar una predicción de precios de ventas de propiedades inmobiliarias, en base al problema publicado en la plataforma Kaggle. Por consiguiente, el objetivo de minería de datos será: la construcción de un modelo inteligible que obtenga una estimación lo más precisa posible del atributo clase *"SalePrice"*, a partir del resto. El nombre de usuario empleado en la plataforma Kaggle es **CDAA17JGFigueira**.

### Carga de liberías y datos

Previante a realizar un análisis de los datos, se procede a cargar las liberías de R requeridas para su realización. Así mismo, se cargan los conjuntos de datos de entrenamiento y tests.

```
required_packages <- c("ggplot2", "dplyr", "caret", "kernlab",
    "glmnet", "xgboost", "data.table", "Metrics", "cowplot",
    "caretEnsemble")
new_packages <- required_packages[!(required_packages %in% installed.packages()[,
    "Package"])]
if (length(new_packages)) install.packages(new_packages)

library(ggplot2)
library(dplyr)
library(caret)
library(kernlab)
library(glmnet)
library(xgboost)
library(data.table)
library(Metrics)
library(cowplot)
library(caretEnsemble)

SEED <- 12345
train <- read.csv("data/train.csv")
test <- read.csv("data/test.csv")
```

## Inspección de los datos

### Análisis preliminar

Como primer paso del proceso, se procede a realizar un análisis de los datos. Para ello, es necesario conocer las dimensiones del conjunto de entrenamiento.

```
dim(train)
```

```
## [1] 1460   81
```

```
dim(test)
```

```
## [1] 1459   80
```

Se observa que el número de instancias del conjunto de entrenamiento contiene prácticamente el mismo número de instancias que el conjunto de test. Sin embargo, este último contiene una variable menos, que se corresponde con la ausencia de variable clase *SalePrice*.

Teniendo en cuenta que en el futuro se harán distintas modificaciones sobre los datos y éstas se deberían realizar teniendo en cuenta el conjunto total de los datos, se procede a juntar los dos conjuntos. Para ello, rellenamos con *NA* los valores ausentes del conjunto de test. Por consiguiente, cuando se realice una operación que tenga en cuenta la variable *SalePrice* (por ejemplo: la correlación entre una de las características y la variable clase), sólo se podrá realizar sobre el subconjunto de test.

Para tener actualizadas ambos subconjuntos, se defina la función *updatePartitions*. De esta forma, todas las operaciones que se apliquen al total de los datos, se verán reflejadas en ambas particiones.

```
full.set <- data.frame(data.table::rbindlist(list(train, cbind(test,
    SalePrice = as.integer(NA))), use.names = F, fill = F))
dim(full.set)
```

```
## [1] 2919   81
```

```
updatePartitions <- function() {
    train.size <- nrow(train)
    full.set.size <- nrow(full.set)

    train <<- full.set[c(1:train.size), ]
    test <<- full.set[c((train.size + 1):full.set.size), ]
}
```

A continuacin, procedemos a examinar las variables del dataset y observamos que coinciden con las descritas por la plataforma.

```
str(full.set)
```

```
## 'data.frame':    2919 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : Factor w/ 5 levels "C (all)","FV",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 4 1 1 1 1 4 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "AllPub","NoSeWa": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
```

```
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 8 levels "ClyTile","CompShg",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 2 3 2 3 2 3 4 4 3 3 ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 4 3 4 3 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 3 3 4 3 3 1 3 4 4 ...
##  $ BsmtCond     : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 2 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "Av","Gd","Mn",..: 4 2 3 4 1 4 1 3 4 4 ...
##  $ BsmtFinType1 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 3 1 3 1 3 3 3 1 6 3 ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 6 6 6 6 6 6 6 2 6 6 ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 1 1 1 3 1 1 1 1 3 1 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 5 levels "FuseA","FuseF",..: 5 5 5 5 5 5 5 5 2 5 ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 3 3 4 3 4 4 4 ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 3 7 ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : Factor w/ 5 levels "Ex","Fa","Gd",..: NA 5 5 3 5 NA 3 5 5 5 ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 6 2 2 2 2 6 2 ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : Factor w/ 3 levels "Fin","RFn","Unf": 2 2 2 3 2 3 2 2 3 2 ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 2 3 ...
##  $ GarageCond   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 3 levels "Ex","Fa","Gd": NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence        : Factor w/ 4 levels "GdPrv","GdWo",..: NA NA NA NA NA 3 NA NA NA NA ...
##  $ MiscFeature  : Factor w/ 4 levels "Gar2","Othr",..: NA NA NA NA NA 3 NA 3 NA NA ...
```

```
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

Así mismo, observamos la información relevante del dataset para tener una idea de sus características. Se aprecia presencia de tanto características numéricas continuas, como ordinálisis y categóricas. También se aprecia una gran presencia de valores perdidos ($NA$) en bastantes características.

```
summary(full.set)
```

```
##        Id           MSSubClass        MSZoning     LotFrontage
##  Min.   :   1.0   Min.   : 20.00   C (all):  25   Min.   : 21.00
##  1st Qu.: 730.5   1st Qu.: 20.00   FV     : 139   1st Qu.: 59.00
##  Median :1460.0   Median : 50.00   RH     :  26   Median : 68.00
##  Mean   :1460.0   Mean   : 57.14   RL     :2265   Mean   : 69.31
##  3rd Qu.:2189.5   3rd Qu.: 70.00   RM     : 460   3rd Qu.: 80.00
##  Max.   :2919.0   Max.   :190.00   NA's   :   4   Max.   :313.00
##                                                   NA's   :486
##     LotArea        Street      Alley      LotShape   LandContour
##  Min.   :  1300   Grvl:  12   Grvl: 120   IR1: 968   Bnk: 117
##  1st Qu.:  7478   Pave:2907   Pave:  78   IR2:  76   HLS: 120
##  Median :  9453               NA's:2721   IR3:  16   Low:  60
##  Mean   : 10168                           Reg:1859   Lvl:2622
##  3rd Qu.: 11570
##  Max.   :215245
##
##   Utilities      LotConfig    LandSlope   Neighborhood    Condition1
##  AllPub:2916   Corner : 511   Gtl:2778   NAmes  : 443   Norm   :2511
##  NoSeWa:   1   CulDSac: 176   Mod: 125   CollgCr: 267   Feedr  : 164
##  NA's   :   2   FR2    :  85   Sev:  16   OldTown: 239   Artery :  92
##                FR3    :  14              Edwards: 194   RRAn   :  50
##                Inside :2133              Somerst: 182   PosN   :  39
##                                          NridgHt: 166   RRAe   :  28
##                                          (Other):1428   (Other):  35
##    Condition2     BldgType      HouseStyle    OverallQual
##  Norm   :2889   1Fam  :2425   1Story :1471   Min.   : 1.000
##  Feedr  :  13   2fmCon:  62   2Story : 872   1st Qu.: 5.000
##  Artery :   5   Duplex: 109   1.5Fin : 314   Median : 6.000
##  PosA   :   4   Twnhs :  96   SLvl   : 128   Mean   : 6.089
##  PosN   :   4   TwnhsE: 227   SFoyer :  83   3rd Qu.: 7.000
##  RRNn   :   2                 2.5Unf :  24   Max.   :10.000
##  (Other):   2                 (Other):  27
##   OverallCond      YearBuilt     YearRemodAdd    RoofStyle
##  Min.   :1.000   Min.   :1872   Min.   :1950   Flat   :  20
##  1st Qu.:5.000   1st Qu.:1954   1st Qu.:1965   Gable  :2310
##  Median :5.000   Median :1973   Median :1993   Gambrel:  22
##  Mean   :5.565   Mean   :1971   Mean   :1984   Hip    : 551
##  3rd Qu.:6.000   3rd Qu.:2001   3rd Qu.:2004   Mansard:  11
##  Max.   :9.000   Max.   :2010   Max.   :2010   Shed   :   5
##
##     RoofMatl     Exterior1st    Exterior2nd     MasVnrType
##  CompShg:2876   VinylSd:1025   VinylSd:1014   BrkCmn :  25
```

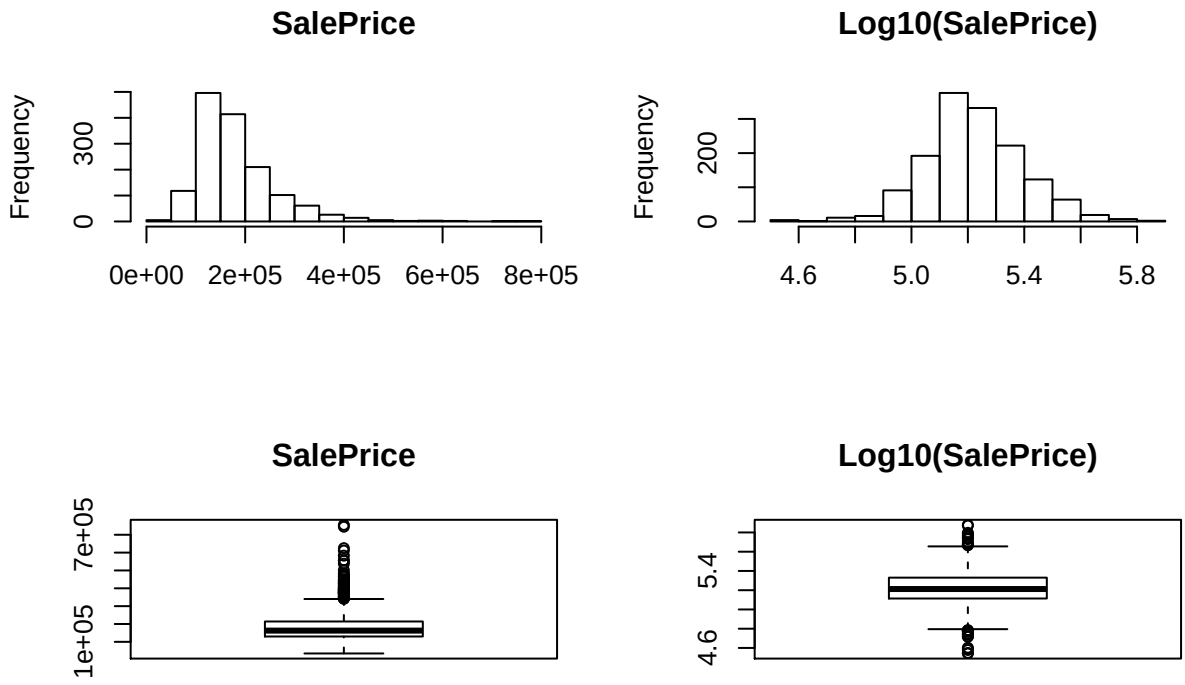
```
##  Tar&Grv:  23   MetalSd: 450   MetalSd: 447   BrkFace: 879
##  WdShake:   9   HdBoard: 442   HdBoard: 406   None   :1742
##  WdShngl:   7   Wd Sdng: 411   Wd Sdng: 391   Stone  : 249
##  ClyTile:   1   Plywood: 221   Plywood: 270   NA's   :  24
##  Membran:   1   (Other): 369   (Other): 390
##  (Other):   2   NA's   :   1   NA's   :   1
##    MasVnrArea     ExterQual ExterCond  Foundation   BsmtQual
##  Min.   :   0.0   Ex: 107   Ex:  12   BrkTil: 311   Ex  : 258
##  1st Qu.:   0.0   Fa:  35   Fa:  67   CBlock:1235   Fa  :  88
##  Median :   0.0   Gd: 979   Gd: 299   PConc :1308   Gd  :1209
##  Mean   : 102.2   TA:1798   Po:   3   Slab  :  49   TA  :1283
##  3rd Qu.: 164.0             TA:2538   Stone :  11   NA's:  81
##  Max.   :1600.0                       Wood  :   5
##  NA's   :23
##  BsmtCond    BsmtExposure BsmtFinType1   BsmtFinSF1     BsmtFinType2
##  Fa  : 104   Av  : 418    ALQ :429     Min.   :   0.0   ALQ :  52
##  Gd  : 122   Gd  : 276    BLQ :269     1st Qu.:   0.0   BLQ :  68
##  Po  :   5   Mn  : 239    GLQ :849     Median : 368.5   GLQ :  34
##  TA  :2606   No  :1904    LwQ :154     Mean   : 441.4   LwQ :  87
##  NA's:  82   NA's:  82    Rec :288     3rd Qu.: 733.0   Rec : 105
##                           Unf :851     Max.   :5644.0   Unf :2493
##                           NA's: 79     NA's   :1        NA's:  80
##    BsmtFinSF2        BsmtUnfSF       TotalBsmtSF      Heating
##  Min.   :   0.00   Min.   :   0.0   Min.   :   0.0   Floor:   1
##  1st Qu.:   0.00   1st Qu.: 220.0   1st Qu.: 793.0   GasA :2874
##  Median :   0.00   Median : 467.0   Median : 989.5   GasW :  27
##  Mean   :  49.58   Mean   : 560.8   Mean   :1051.8   Grav :   9
##  3rd Qu.:   0.00   3rd Qu.: 805.5   3rd Qu.:1302.0   OthW :   2
##  Max.   :1526.00   Max.   :2336.0   Max.   :6110.0   Wall :   6
##  NA's   :1         NA's   :1        NA's   :1
##  HeatingQC CentralAir Electrical     X1stFlrSF      X2ndFlrSF
##  Ex:1493   N: 196     FuseA: 188   Min.   : 334   Min.   :   0.0
##  Fa:  92   Y:2723     FuseF:  50   1st Qu.: 876   1st Qu.:   0.0
##  Gd: 474              FuseP:   8   Median :1082   Median :   0.0
##  Po:   3              Mix  :   1   Mean   :1160   Mean   : 336.5
##  TA: 857              SBrkr:2671   3rd Qu.:1388   3rd Qu.: 704.0
##                       NA's :   1   Max.   :5095   Max.   :2065.0
##
##   LowQualFinSF        GrLivArea     BsmtFullBath     BsmtHalfBath
##  Min.   :   0.000   Min.   : 334   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:   0.000   1st Qu.:1126   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :   0.000   Median :1444   Median :0.0000   Median :0.00000
##  Mean   :   4.694   Mean   :1501   Mean   :0.4299   Mean   :0.06136
##  3rd Qu.:   0.000   3rd Qu.:1744   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :1064.000   Max.   :5642   Max.   :3.0000   Max.   :2.00000
##                                    NA's   :2        NA's   :2
##     FullBath        HalfBath       BedroomAbvGr   KitchenAbvGr
##  Min.   :0.000   Min.   :0.0000   Min.   :0.00   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:2.00   1st Qu.:1.000
##  Median :2.000   Median :0.0000   Median :3.00   Median :1.000
##  Mean   :1.568   Mean   :0.3803   Mean   :2.86   Mean   :1.045
##  3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:3.00   3rd Qu.:1.000
##  Max.   :4.000   Max.   :2.0000   Max.   :8.00   Max.   :3.000
##
```

```
##  KitchenQual  TotRmsAbvGrd      Functional     Fireplaces     FireplaceQu
##  Ex  : 205   Min.   : 2.000   Typ    :2717   Min.   :0.0000   Ex  :  43
##  Fa  :  70   1st Qu.: 5.000   Min2   :  70   1st Qu.:0.0000   Fa  :  74
##  Gd  :1151   Median : 6.000   Min1   :  65   Median :1.0000   Gd  : 744
##  TA  :1492   Mean   : 6.452   Mod    :  35   Mean   :0.5971   Po  :  46
##  NA's:   1   3rd Qu.: 7.000   Maj1   :  19   3rd Qu.:1.0000   TA  : 592
##              Max.   :15.000   (Other):  11   Max.   :4.0000   NA's:1420
##                               NA's   :   2
##    GarageType    GarageYrBlt   GarageFinish   GarageCars
##  2Types :  23   Min.   :1895   Fin : 719    Min.   :0.000
##  Attchd :1723   1st Qu.:1960   RFn : 811    1st Qu.:1.000
##  Basment:  36   Median :1979   Unf :1230    Median :2.000
##  BuiltIn: 186   Mean   :1978   NA's: 159    Mean   :1.767
##  CarPort:  15   3rd Qu.:2002                3rd Qu.:2.000
##  Detchd : 779   Max.   :2207                Max.   :5.000
##  NA's   : 157   NA's   :159                 NA's   :1
##    GarageArea     GarageQual  GarageCond  PavedDrive   WoodDeckSF
##  Min.   :   0.0   Ex  :   3   Ex  :   3   N: 216     Min.   :   0.00
##  1st Qu.: 320.0   Fa  : 124   Fa  :  74   P:  62     1st Qu.:   0.00
##  Median : 480.0   Gd  :  24   Gd  :  15   Y:2641     Median :   0.00
##  Mean   : 472.9   Po  :   5   Po  :  14              Mean   :  93.71
##  3rd Qu.: 576.0   TA  :2604   TA  :2654              3rd Qu.: 168.00
##  Max.   :1488.0   NA's: 159   NA's: 159              Max.   :1424.00
##  NA's   :1
##   OpenPorchSF     EnclosedPorch      X3SsnPorch       ScreenPorch
##  Min.   :  0.00   Min.   :   0.0   Min.   :  0.000   Min.   :  0.00
##  1st Qu.:  0.00   1st Qu.:   0.0   1st Qu.:  0.000   1st Qu.:  0.00
##  Median : 26.00   Median :   0.0   Median :  0.000   Median :  0.00
##  Mean   : 47.49   Mean   :  23.1   Mean   :  2.602   Mean   : 16.06
##  3rd Qu.: 70.00   3rd Qu.:   0.0   3rd Qu.:  0.000   3rd Qu.:  0.00
##  Max.   :742.00   Max.   :1012.0   Max.   :508.000   Max.   :576.00
##
##     PoolArea        PoolQC       Fence      MiscFeature    MiscVal
##  Min.   :  0.000   Ex  :   4   GdPrv: 118   Gar2:   5   Min.   :    0.00
##  1st Qu.:  0.000   Fa  :   2   GdWo : 112   Othr:   4   1st Qu.:    0.00
##  Median :  0.000   Gd  :   4   MnPrv: 329   Shed:  95   Median :    0.00
##  Mean   :  2.252   NA's:2909   MnWw :  12   TenC:   1   Mean   :   50.83
##  3rd Qu.:  0.000               NA's :2348   NA's:2814   3rd Qu.:    0.00
##  Max.   :800.000                                       Max.   :17000.00
##
##      MoSold           YrSold        SaleType    SaleCondition
##  Min.   : 1.000   Min.   :2006   WD     :2525   Abnorml: 190
##  1st Qu.: 4.000   1st Qu.:2007   New    : 239   AdjLand:  12
##  Median : 6.000   Median :2008   COD    :  87   Alloca :  24
##  Mean   : 6.213   Mean   :2008   ConLD  :  26   Family :  46
##  3rd Qu.: 8.000   3rd Qu.:2009   CWD    :  12   Normal :2402
##  Max.   :12.000   Max.   :2010   (Other):  29   Partial: 245
##                                  NA's   :   1
##    SalePrice
##  Min.   : 34900
##  1st Qu.:129975
##  Median :163000
##  Mean   :180921
##  3rd Qu.:214000
```

```
##  Max.   :755000
##  NA's   :1459
```

Otra característica a destacar es la elevada diferencia entre los valores de la media (180921) y la mediana (163000) de la variable clase *SalePrice* ($17921 de diferencia). Esto puede apuntar a una desviación por la presencia de valores anómalos.

```
par(mfrow = c(2, 2))

hist(train$SalePrice, main = "SalePrice", xlab = "")
hist(log10(train$SalePrice), main = "Log10(SalePrice)", xlab = "")

boxplot(train$SalePrice, main = "SalePrice")
boxplot(log10(train$SalePrice), main = "Log10(SalePrice)")
```




```
par(mfrow = c(1, 1))
```

Se ha podido comprobar la clara desviación de los datos y como la aplicación de logaritmos ayuda en su corrección. De esta forma, se aplicará esta modificación para eliminar esta asimetria (o sesgo) en las operaciones que utilicen esta variable.

## Análisis de valores perdidos

Dado que se ha advertido de una elevada presencia de valores perdidos, se procede a la obtener una información clara de los valores perdidos que contiene cada característica. Para ello se define la función *getLostValuesStats*, de esta forma se podrá comprobar recurrentemente, de una forma rápida, cuántos valores perdidos quedan por tratar.

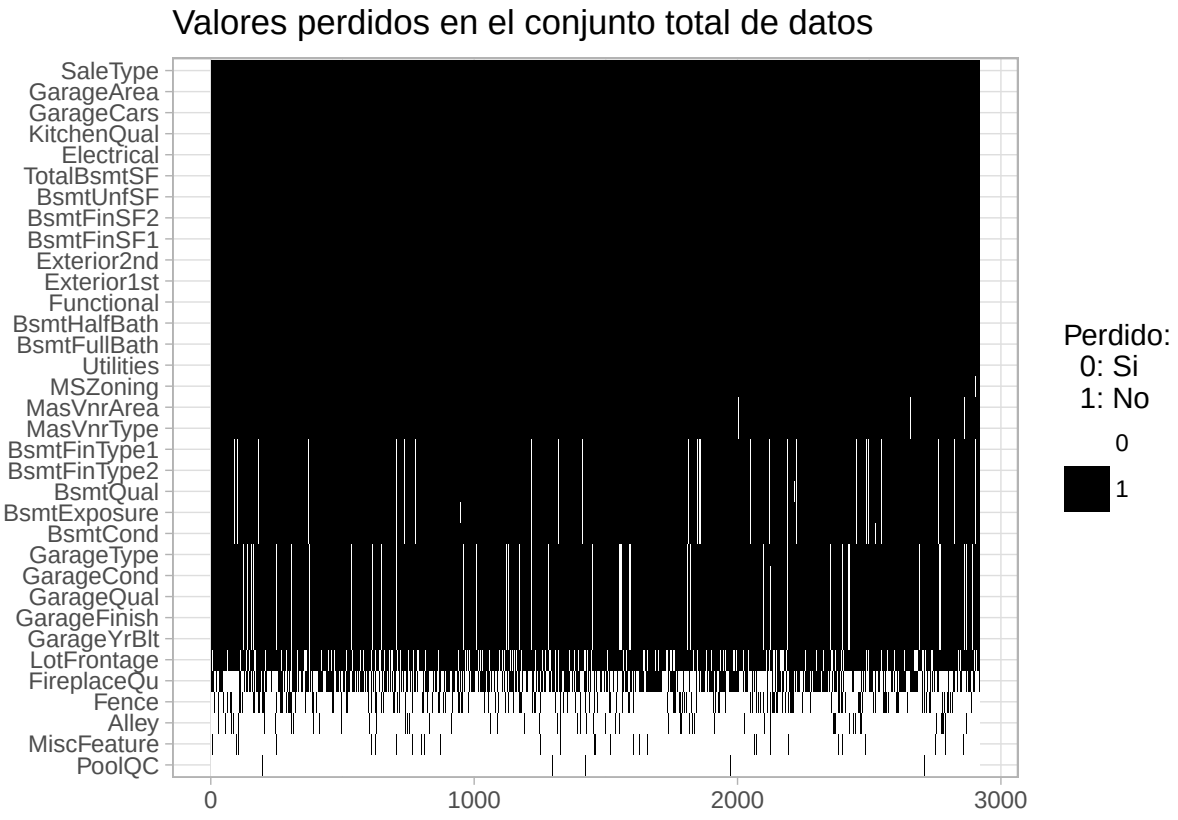
```
getLostValuesStats <- function() {
    lost.count <- colSums(sapply(select(full.set, -SalePrice),
        is.na))
    lost.count <- subset(lost.count, lost.count > 0)
    lost.percentage <- (lost.count/nrow(full.set)) * 100

    return(data.frame(lost.count, lost.percentage))
}

getLostValuesStats()
```

```
##              lost.count lost.percentage
## MSZoning              4      0.13703323
## LotFrontage         486     16.64953751
## Alley              2721     93.21685509
## Utilities             2      0.06851662
## Exterior1st           1      0.03425831
## Exterior2nd           1      0.03425831
## MasVnrType           24      0.82219938
## MasVnrArea           23      0.78794108
## BsmtQual             81      2.77492292
## BsmtCond             82      2.80918123
## BsmtExposure         82      2.80918123
## BsmtFinType1         79      2.70640630
## BsmtFinSF1            1      0.03425831
## BsmtFinType2         80      2.74066461
## BsmtFinSF2            1      0.03425831
## BsmtUnfSF             1      0.03425831
## TotalBsmtSF           1      0.03425831
## Electrical            1      0.03425831
## BsmtFullBath          2      0.06851662
## BsmtHalfBath          2      0.06851662
## KitchenQual           1      0.03425831
## Functional            2      0.06851662
## FireplaceQu        1420     48.64679685
## GarageType          157      5.37855430
## GarageYrBlt         159      5.44707091
## GarageFinish        159      5.44707091
## GarageCars            1      0.03425831
## GarageArea            1      0.03425831
## GarageQual          159      5.44707091
## GarageCond          159      5.44707091
## PoolQC             2909     99.65741692
## Fence              2348     80.43850634
## MiscFeature        2814     96.40287770
## SaleType              1      0.03425831
```

Ahora se procede a obtener una visual del estado del conjunto de datos, en cuanto lo que se refiere a la inclusión de este tipo de valores.

```
lost.values.count <- full.set[, colSums(is.na(select(full.set,
    -SalePrice))) > 0]

is.lost.value <- as.data.frame(ifelse(is.na(lost.values.count),
    0, 1))
```

```
is.lost.value <- is.lost.value[, order(colSums(is.lost.value))]

is.lost.value.grid <- expand.grid(list(x = 1:nrow(is.lost.value),
    y = colnames(is.lost.value)))
is.lost.value.grid$m <- as.vector(as.matrix(is.lost.value))
is.lost.value.grid <- data.frame(x = unlist(is.lost.value.grid$x),
    y = unlist(is.lost.value.grid$y), m = unlist(is.lost.value.grid$m))

ggplot2::ggplot(is.lost.value.grid) + geom_tile(aes(x = x, y = y,
    fill = factor(m))) + scale_fill_manual(values = c("white",
    "black"), name = "Perdido:\n  0: Si\n  1: No") + theme_light() +
    ylab("") + xlab("") + ggtitle("Valores perdidos en el conjunto total de datos")
```



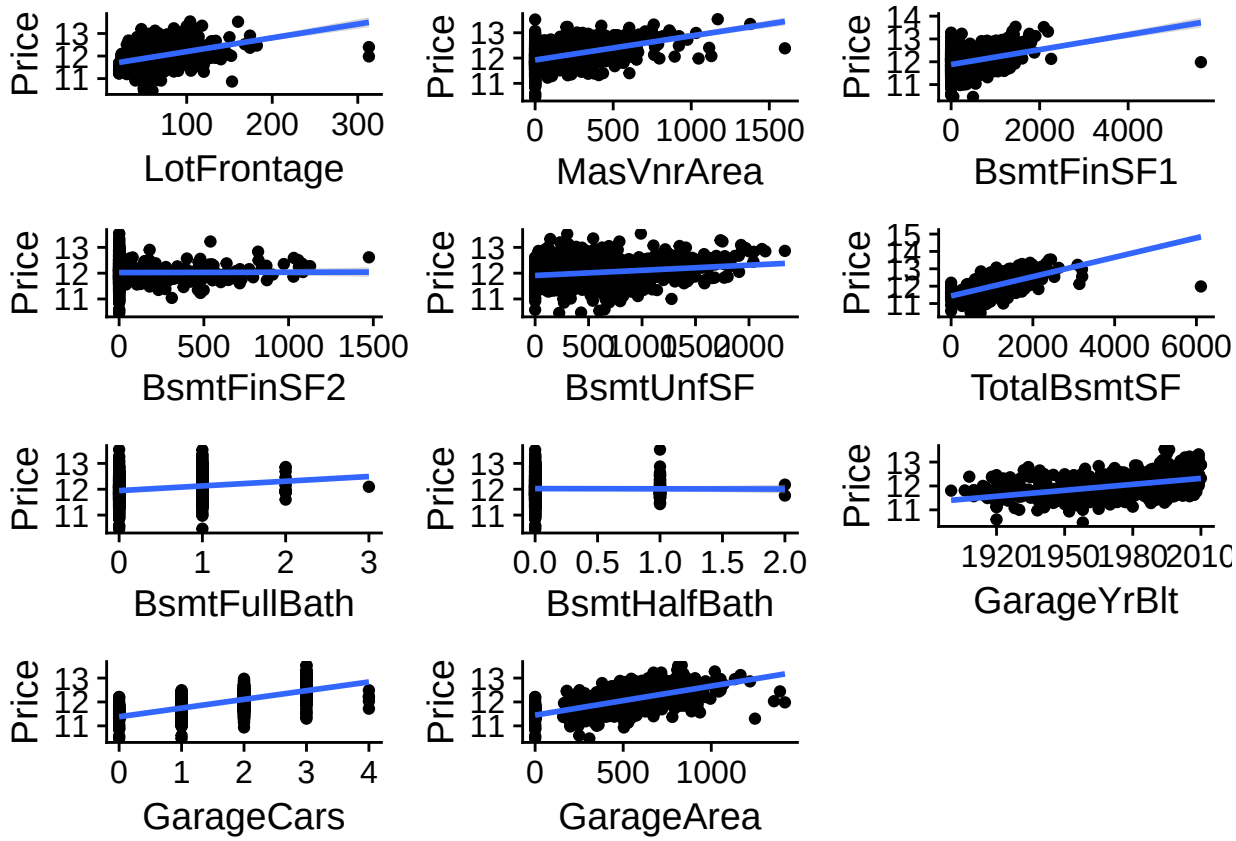

## Tratamiento de los valores perdidos

Dada la heterogeneidad de los valores faltantes, se procederá a un análisis muy pormenorizado. En primer lugar, se tratarán las características numéricas con valores faltantes más representativos y luego se analizarán las variables categóricas.

En primer lugar, se procede a examinar la distribución de las variables numéricas, que contengan valores perdidos, con respecto a la variable *Log(SalePrice)*. Se puede observar que, además de la presencia de valores perdidos, hay algunas variables que contienen un elevado número de entradas con valor 0 (por ejemplo: *MasVnrArea*).

```
lost.values.features <- rownames(getLostValuesStats())
numeric.features <- names(train)[which(sapply(train, is.numeric))]
lost.values.features.numeric <- dplyr::intersect(numeric.features,
    lost.values.features)

plots <- lapply(lost.values.features.numeric, function(feature) {
    ggplot(data = train, aes(x = train[, feature], y = log(train$SalePrice))) +
        geom_point() + geom_smooth(method = "lm") + xlab(label = feature) +
        ylab(label = "Price")
})

cowplot::plot_grid(plotlist = plots, ncol = 3)
```

```
## Warning: Removed 259 rows containing non-finite values (stat_smooth).
## Warning: Removed 259 rows containing missing values (geom_point).
## Warning: Removed 8 rows containing non-finite values (stat_smooth).
## Warning: Removed 8 rows containing missing values (geom_point).
## Warning: Removed 81 rows containing non-finite values (stat_smooth).
## Warning: Removed 81 rows containing missing values (geom_point).
```



**GarageYrBlt**

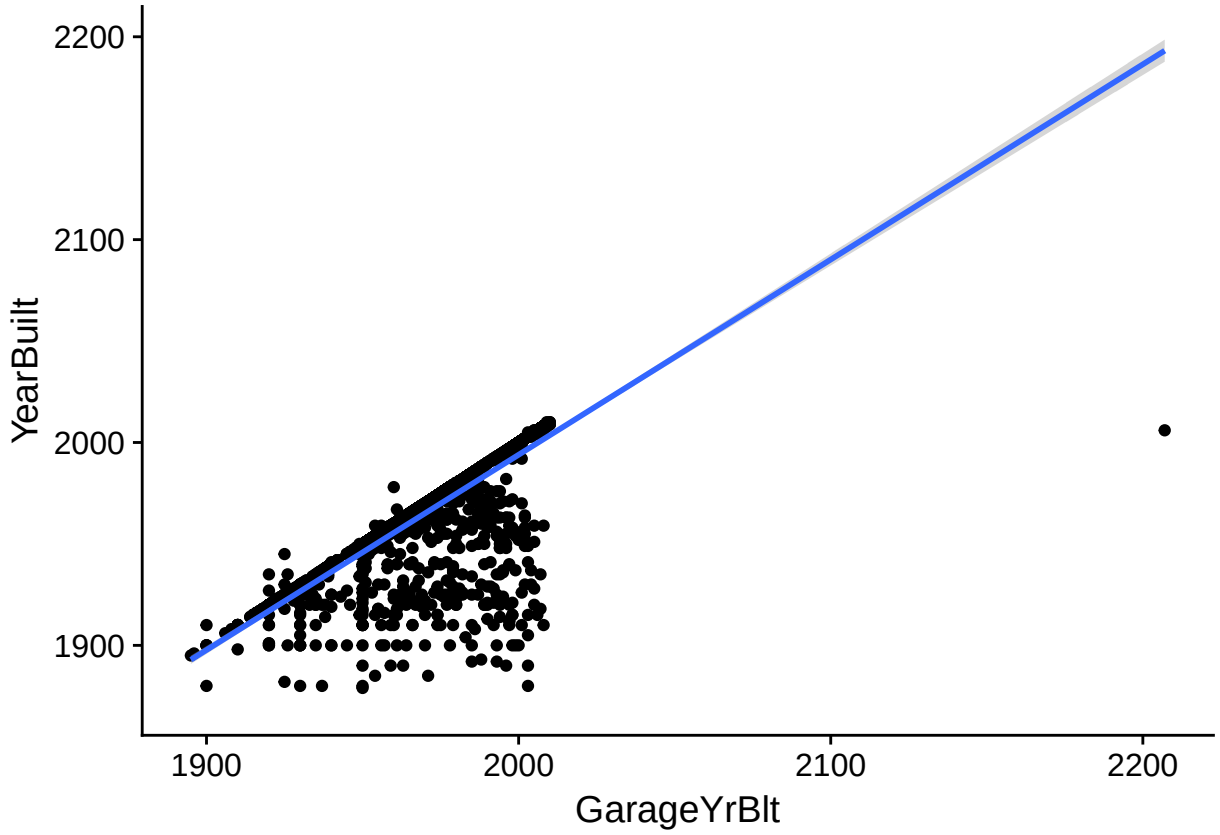
Se aprecia que *GarageYrBlt* (año de construcción del garage) es una propidad que, lógicamente, está muy relacionada con *YearBuilt* (año de construcción). En general, se puede decir que *GarageYrBlt* tiende a ser igual a *YearBuilt*. Por consiguiente, en los valores perdidos de GarageYrBlt, se procede a asígnar el correspondiente valor de YearBuilt.

Así mismo, en la gráfica se observa la existencia de una inconsistencia, dado que una de la instancias toma valor 2207, cuando no es posible que contenga dicho año. Se sobreentiende que el valor que debería contener es 2007.

```
ggplot(data = full.set, aes(x = GarageYrBlt, y = YearBuilt)) +
    geom_point() + geom_smooth(method = "lm")
```

```
## Warning: Removed 159 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 159 rows containing missing values (geom_point).
```




```
full.set$GarageYrBlt[full.set$GarageYrBlt == 2207] <- 2007
selected <- is.na(full.set$GarageYrBlt)
full.set$GarageYrBlt[selected] <- full.set$YearBuilt[selected]

updatePartitions()  # Se actualizan los subconjuntos de entrenamiento y test
```

**LotFrontage**

Por lógica, se puede decir que el área de la propiedad (*LotArea*) guarda relación con la longitud de la fachada (*LotFrontage*). Para confirmarlo, comprobamos la correlación entre ellas:

```
cor(full.set$LotFrontage, full.set$LotArea, use = "complete.obs")
```

```
## [1] 0.4898956
```

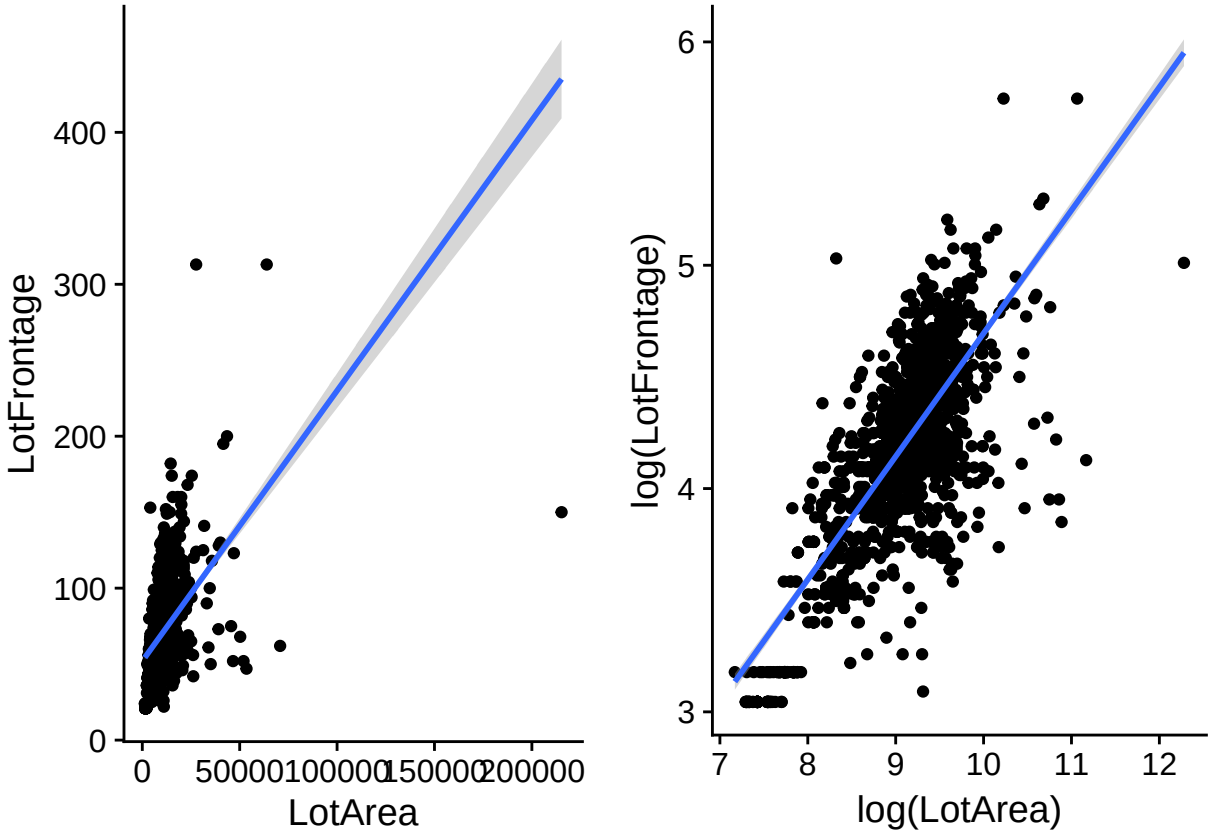
```
cor(log(full.set$LotFrontage), log(full.set$LotArea), use = "complete.obs")
```

```
## [1] 0.7662858
```

Y visualizamos su relación:

```
plotLotRelation <- ggplot(data = full.set, aes(x = LotArea, y = LotFrontage)) +
    geom_point() + geom_smooth(method = "lm")

plotLogLotRelation <- ggplot(data = full.set, aes(x = log(LotArea),
    y = log(LotFrontage))) + geom_point() + geom_smooth(method = "lm")

cowplot::plot_grid(plotLotRelation, plotLogLotRelation, ncol = 2)
```

```
## Warning: Removed 486 rows containing non-finite values (stat_smooth).
## Warning: Removed 486 rows containing missing values (geom_point).
## Warning: Removed 486 rows containing non-finite values (stat_smooth).
## Warning: Removed 486 rows containing missing values (geom_point).
```



Se puede confirmar que existe una importante correlación entre *LotFrontage* y *LotArea*. Dado que estas dos propiedades están relacionadas, seguramente una de ellas sea desechada en el proceso de selección de variables. Independientemente de ello, en este paso sustituiremos los valores de *LotFrontage*, por la mediana de los valores existentes.

```
selected <- is.na(full.set$LotFrontage)
full.set$LotFrontage[selected] <- mean(full.set$LotFrontage[!selected])

updatePartitions()
```

Finalmente, observamos la correlaciones después de realizar las modificaciones y apreciamos como se han modificado pero siguen conservando la misma tendencia.

```
cor(full.set$LotFrontage, full.set$LotArea, use = "complete.obs")
```

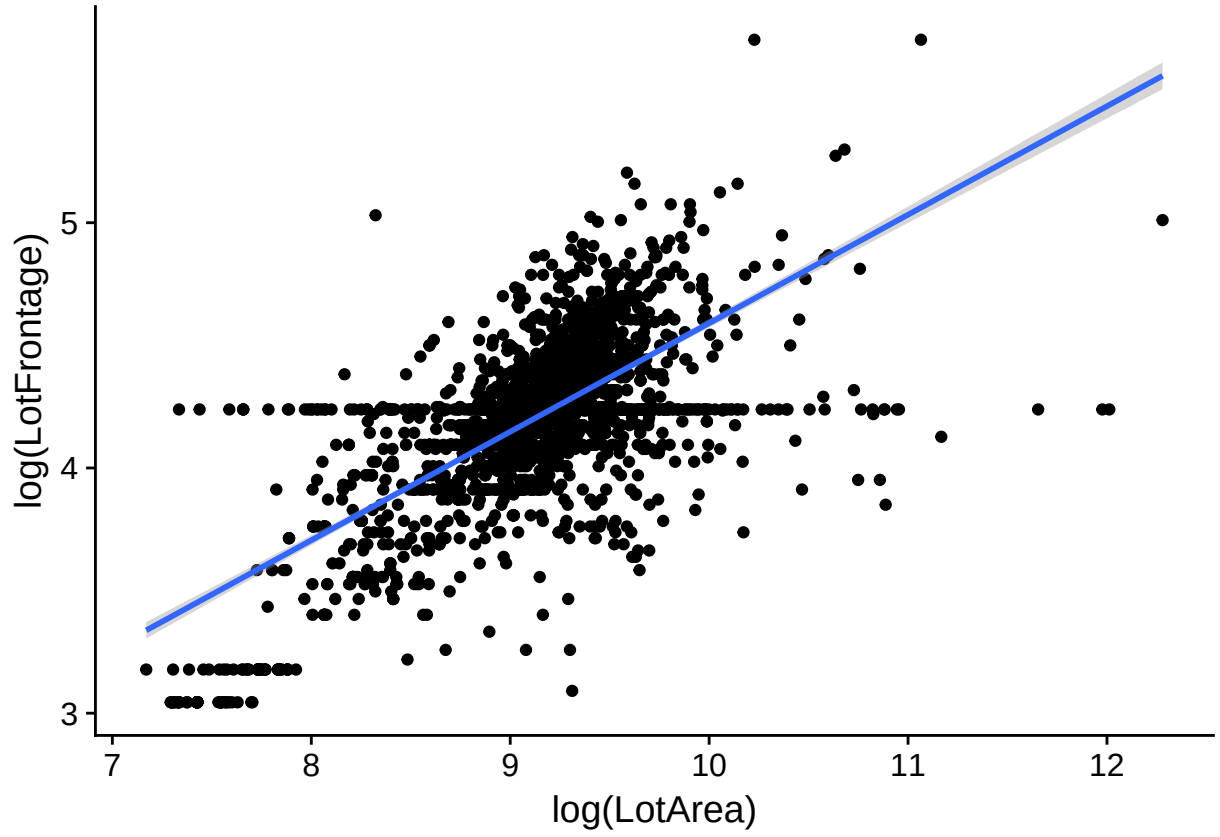
```
## [1] 0.364382
```

```
cor(log(full.set$LotFrontage), log(full.set$LotArea), use = "complete.obs")
```

```
## [1] 0.6894001
```

Visualmente se aprecia que la correlación continúa siendo similar, después de tratar los valores perdidos en *LotFrontage*.

```
ggplot(data = full.set, aes(x = log(LotArea), y = log(LotFrontage))) +
    geom_point() + geom_smooth(method = "lm")
```

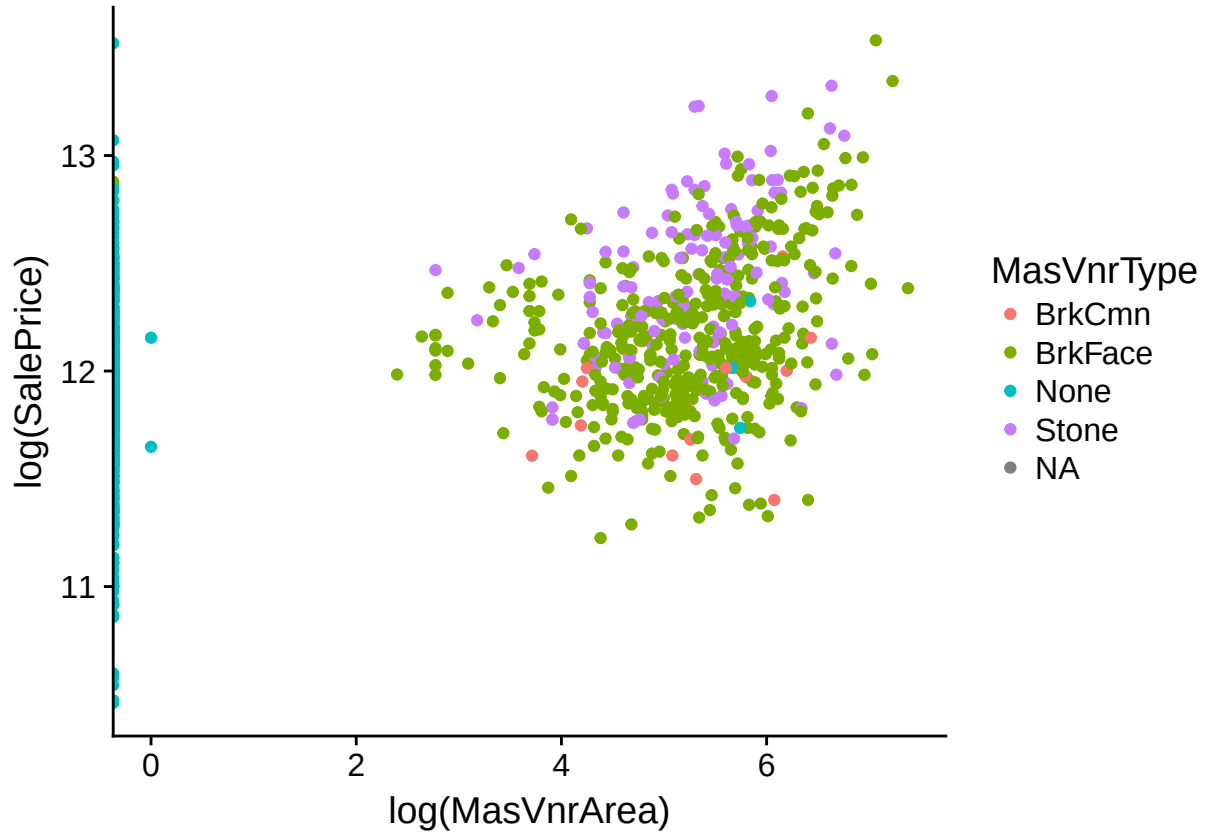


**MasVnrArea**

En esta característica (área de chapa de mampostería), existen una gran cantidad de entradas con valor 0. Esto, seguramente, se deba a la carencia de chapado en la vivienda. Para confirmarlo, visualizamos el cruce entre esta variable y el precio de venta (*SalePrice*), clasificando las intancias según el tipo de chapado de mampostería que tiene la vivienda.

```
ggplot2::qplot(data = train, x = log(MasVnrArea), y = log(SalePrice),
    col = MasVnrType)
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```




También observamos que en ambas variables los valores perdidos (8) forman parte, prácticamente, de los mismos ejemplos:

```
full.set$Id[is.na(full.set$MasVnrArea)]
```

```
##  [1]  235  530  651  937  974  978 1244 1279 1692 1707 1883 1993 2005 2042
## [15] 2312 2326 2341 2350 2369 2593 2658 2687 2863
```

```
full.set$Id[is.na(full.set$MasVnrType)]
```

```
##  [1]  235  530  651  937  974  978 1244 1279 1692 1707 1883 1993 2005 2042
## [15] 2312 2326 2341 2350 2369 2593 2611 2658 2687 2863
```

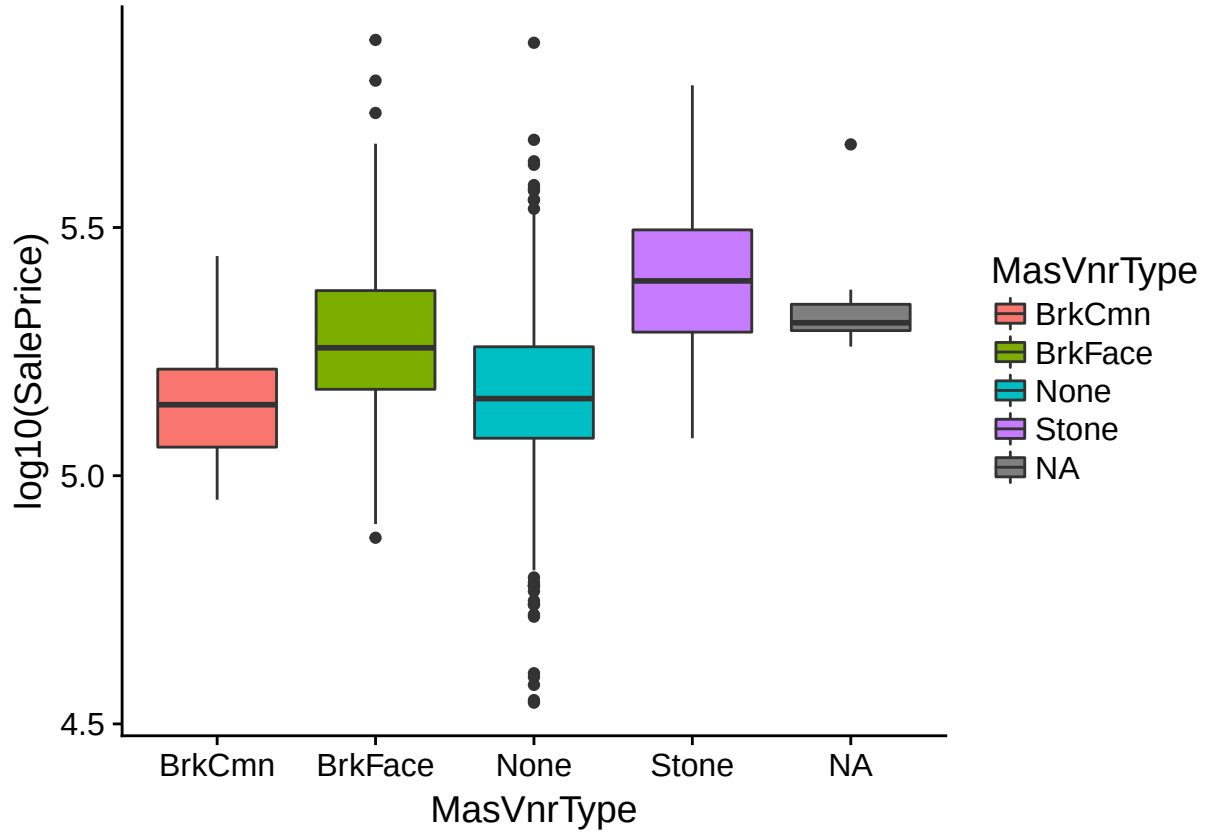
Se elimina la caraterística *MasVnrArea*, ya que las entradas con valor 0, por ser del tipo "None", hacen que la información desprendida de la variable esté deformada.

```
full.set <- dplyr::select(full.set, -MasVnrArea)
```

```
updatePartitions()
```

Ahora se deben tratar los valores perdidos de *MasVnrType*. Para ello, observamos *MasVnrType* en relación a *SalePrice* para entender su distribución.

```
qplot(data = train, x = MasVnrType, y = log10(SalePrice), geom = c("boxplot"),
    fill = MasVnrType)
```

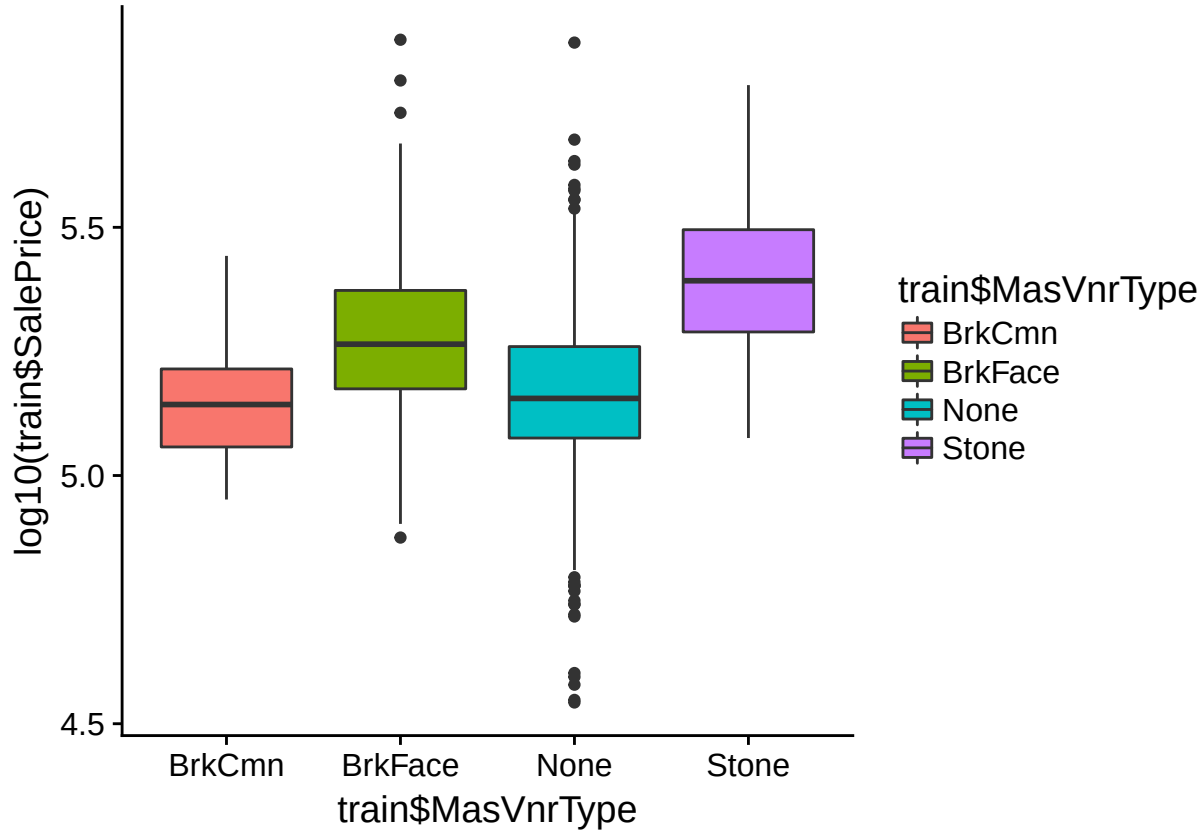



Asignamos a los valores perdidos el tipo *BrkFace*, por mayor proximidad de sus medias. Aunque también se les podría asignar el tipo "Stone".

```
full.set$MasVnrType[is.na(full.set$MasVnrType)] <- "BrkFace"

updatePartitions()
```

Finalmente, observamos la distribución resultante en las categorías de *MasVnrType* en relación a *SalePrice*.

```
qplot(data = train, x = train$MasVnrType, y = log10(train$SalePrice),
    geom = c("boxplot"), fill = train$MasVnrType)
```




**Tratamiento de variables categóricas**

Las siguientes caraterísticas con valores perdidos se corresponden con aquellas que contienen entradas en las que hay una ausencia de la propiedad a la que representan. Por ejemplo, la propiedad *PoolQC* representa la calidad de la piscina. Pero es obvio que en aquellas propiedades en las haya una ausencia de piscina, será inviable representar su calidad. Por consiguiente, se le asignarán el tipo "None" a aquellos valores ausentes (). Las caracteríasticas que contienen este tipo de valores perdidos son: *PoolQC*, *MiscFeature*, *Alley*, *Fence*, *FireplaceQu*, *GarageCond*, *GarageFinish*, *GarageQual*, *GarageType*, *BsmtCond*, *BsmtExposure*, *BsmtFinType1*, *BsmtFinType2* y *BsmtQual*.

```
categorical.features <- c("PoolQC", "MiscFeature", "Alley", "Fence",
    "FireplaceQu", "GarageCond", "GarageFinish", "GarageQual",
    "GarageType", "BsmtCond", "BsmtExposure", "BsmtFinType1",
    "BsmtFinType2", "BsmtQual")

for (feature in categorical.features) {
    if (is.factor(full.set[, feature])) {
        full.set[, feature] <- as.character(full.set[, feature])
        full.set[, feature][which(is.na(full.set[, feature]))] <- "None"
        full.set[, feature] <- factor(full.set[, feature])
    }
}

updatePartitions()
```

### Electrical

Esta característica presenta solamente un valor perdido. Dada la mínima influencia que puede tener, se le asigna la categoría mayoritaria.

```
full.set$Electrical[is.na(full.set$Electrical)] <- as.character(sort(full.set$Electrical,
    decreasing = TRUE)[1])

updatePartitions()
```

### Corrección de valores perdidos en el conjunto de test

Se examinan los valores perdidos que restan el dataset y se aprecia que aún existen algunos datos sin tratar. Aunque estas entradas pertenecen al conjunto de test, se proceden a procesar de la misma forma que se hizo en el conjunto de entrenamiento.

```
getLostValuesStats()
```

```
##              lost.count lost.percentage
## MSZoning              4      0.13703323
## Utilities             2      0.06851662
## Exterior1st           1      0.03425831
## Exterior2nd           1      0.03425831
## BsmtFinSF1            1      0.03425831
## BsmtFinSF2            1      0.03425831
## BsmtUnfSF             1      0.03425831
## TotalBsmtSF           1      0.03425831
## BsmtFullBath          2      0.06851662
## BsmtHalfBath          2      0.06851662
## KitchenQual           1      0.03425831
## Functional            2      0.06851662
## GarageCars            1      0.03425831
## GarageArea            1      0.03425831
## SaleType              1      0.03425831
```

Por una parte, se procesan las características de tipo numérico, asignando la mediana a los valores faltantes.

```
lost.test.numeric.features <- c("BsmtFinSF1", "BsmtFinSF2", "BsmtUnfSF",
    "TotalBsmtSF", "BsmtFullBath", "BsmtHalfBath", "GarageCars",
    "GarageArea")

for (feature in lost.test.numeric.features) {
    full.set[, feature][is.na(full.set[, feature])] <- median(full.set[,
        feature][!is.na(full.set[, feature])])
}

updatePartitions()
```

En cuanto a las variables de tipo nominal, se les asigna la moda de sus valores.

```
lost.test.categorical.features <- c("Exterior1st", "Exterior2nd",
    "Functional", "KitchenQual", "MSZoning", "SaleType", "Utilities")

for (feature in lost.test.categorical.features) {
    full.set[, feature][is.na(full.set[, feature])] <- as.character(sort(full.set[,
        feature], decreasing = T)[1])
```

```
}

updatePartitions()
```

Antes de continuar, comprobamos que no hay valores perdidos en ninguno de los dos conjuntos.

```
getLostValuesStats()
```

```
## [1] lost.count      lost.percentage
## <0 rows> (or 0-length row.names)
```

# Transformación de los datos

En esta sección se procederá a aplicar determinados procesos sobre los datos, tales como: eliminación de variables superfluas, generación de variables, eliminación de intancias, escalado de los datos... de forma que se obtenga una conjunto de datos resultate óptimo para la creación del modelo.

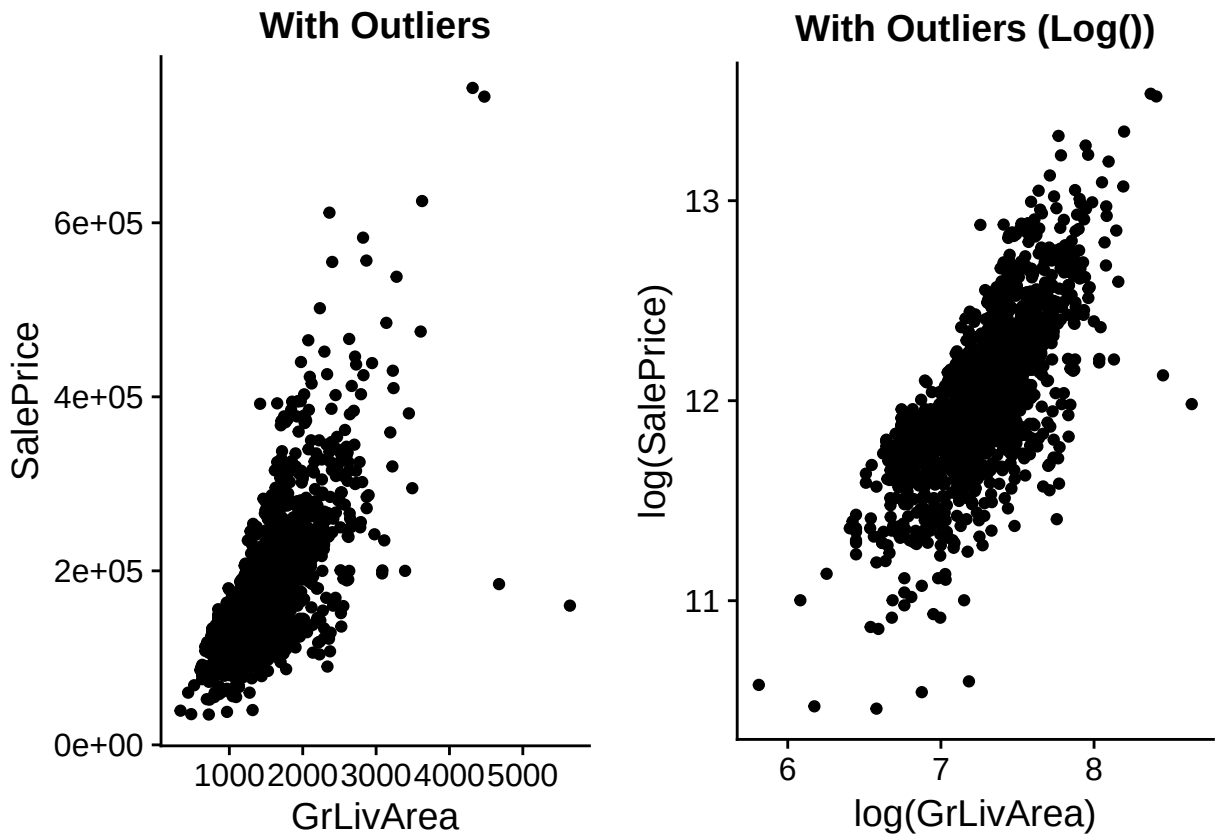
En primer lugar, se procede a eliminar la propiedad *Id* de los conjuntos de entrenamiento y test, debido a su irrelevancia.

```
train.transformed <- dplyr::select(train, -Id)
test.transformed <- dplyr::select(test, -Id)
```

## Tratamiento de outliers

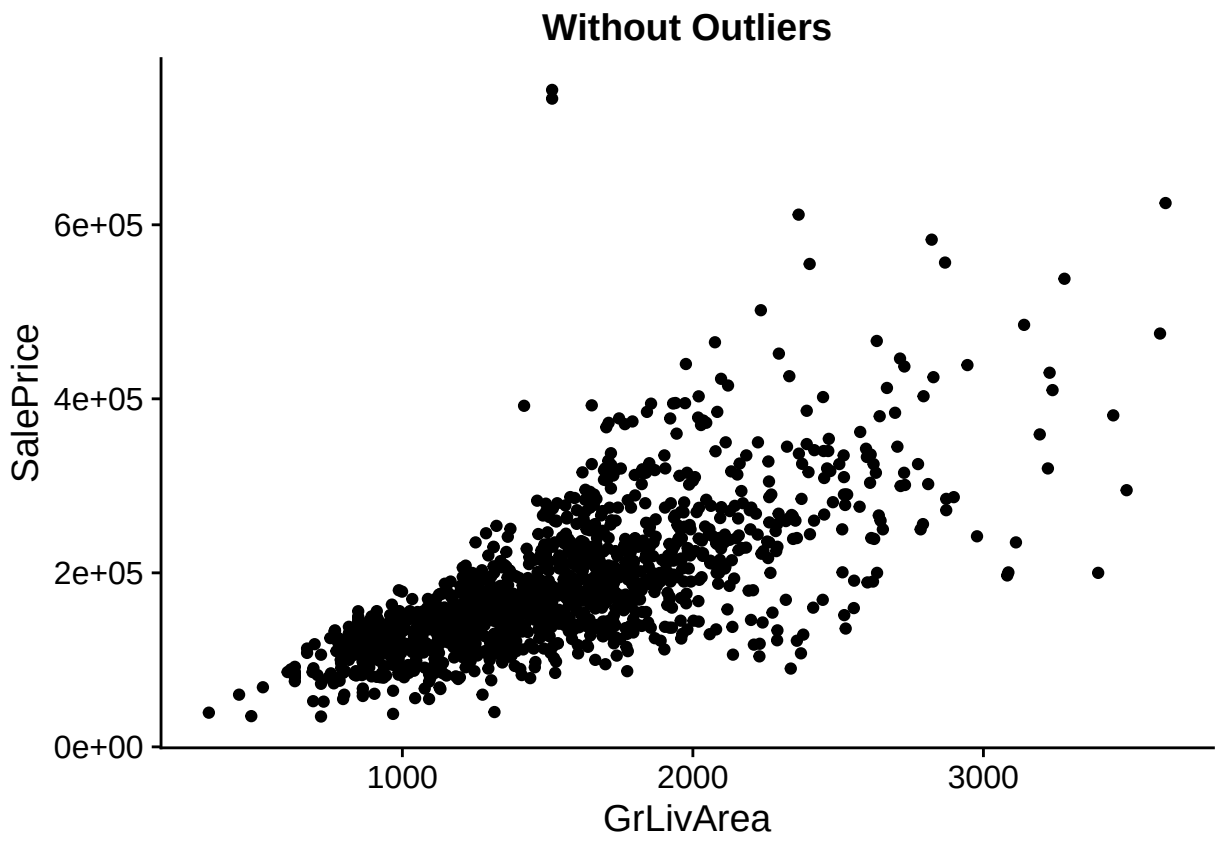
En la fase de análisis de los datos se ha observado como la propiedad *GrLivArea* (area habitable de la vivienda), contiene algunas instancias que se podrían etiquetar como valores anómalos.

```
plot.with.outliers <- ggplot(train.transformed, aes(y = SalePrice,
    x = GrLivArea)) + ggtitle("With Outliers") + geom_point()

plot.with.outliers.log <- ggplot(train.transformed, aes(y = log(SalePrice),
    x = log(GrLivArea))) + ggtitle("With Outliers (Log())") +
    geom_point()

cowplot::plot_grid(plot.with.outliers, plot.with.outliers.log,
    ncol = 2)
```




Se puede aprecicar como hay algunas intancias con valores muy alejados de la distribución. Incluso aplicando logaritmos, toman valores alejados de la distribución. Más adelante se comprobará como esta variable tiene una gran importancia en el model. Para evitar desviaciones, se procede a asignarles la media.

```
GrLivArea.mean <- mean(train.transformed$GrLivArea) %>% as.numeric
train.transformed[train.transformed$GrLivArea > 4000, ]$GrLivArea <- GrLivArea.mean
ggplot(train.transformed, aes(y = SalePrice, x = GrLivArea)) +
    ggtitle("Without Outliers") + geom_point()
```




Se procede a volver a unir ambos subconjuntos, para aplicar conjuntamente las siguientes transformaciones.

```
full.set.transformed <- data.frame(data.table::rbindlist(list(train.transformed,
    test.transformed), use.names = F, fill = F))
```

### Ingeniería de caraterísticas

En este apartado, se procede a la eliminación y generaciòn de características del dataset. Tal y como se ha comentado en el *"Análisis de valores perdidos"*, la característica *LotFrontage* es bastante redundante respecto a la propiedad *LotArea*. Y se ha "deformado" la correlación entre ambas al realizar el tratamiento de valores perdidos. Debido a esto, se procede a eliminar la característica.

Así mismo, se generan las siguientes características, que intuitivamente y tras sucesivas generaciones de modelos ayudan a la generalización del mismo:

- **TotalSF**: superficie de la vivienda, en pies cuadrados. Se genera utilizando las características que referencian la superficie del sótano (*TotalBsmtSF*), primer piso (*X1stFlrSF*) y segundo piso (*X2ndFlrSF*).
- **Age**: edad efectiva de la vivienda. Se genera sustayendo el año de remodelación de la vivienda (*YearRemodAdd*) al año de venta de la misma (*YrSold*).
- **TotalProch**: area total de porche, en pies cuatrados. Se genera utlizando las vararíable que contienene información referente a las superficise de distintos tipos de porches (*EnclosedPorch*, *ScreenPorch* y *X3SsnPorch*).

```
full.set.transformed <- dplyr::select(full.set.transformed, -LotFrontage)

full.set.transformed$TotalSF = full.set.transformed$TotalBsmtSF +
    full.set.transformed$X1stFlrSF + full.set.transformed$X2ndFlrSF
```

```
full.set.transformed$Age <- full.set.transformed$YrSold - full.set.transformed$YearRemodAdd

full.set.transformed$TotalProch <- full.set.transformed$EnclosedPorch +
    full.set.transformed$ScreenPorch + full.set.transformed$X3SsnPorch
```

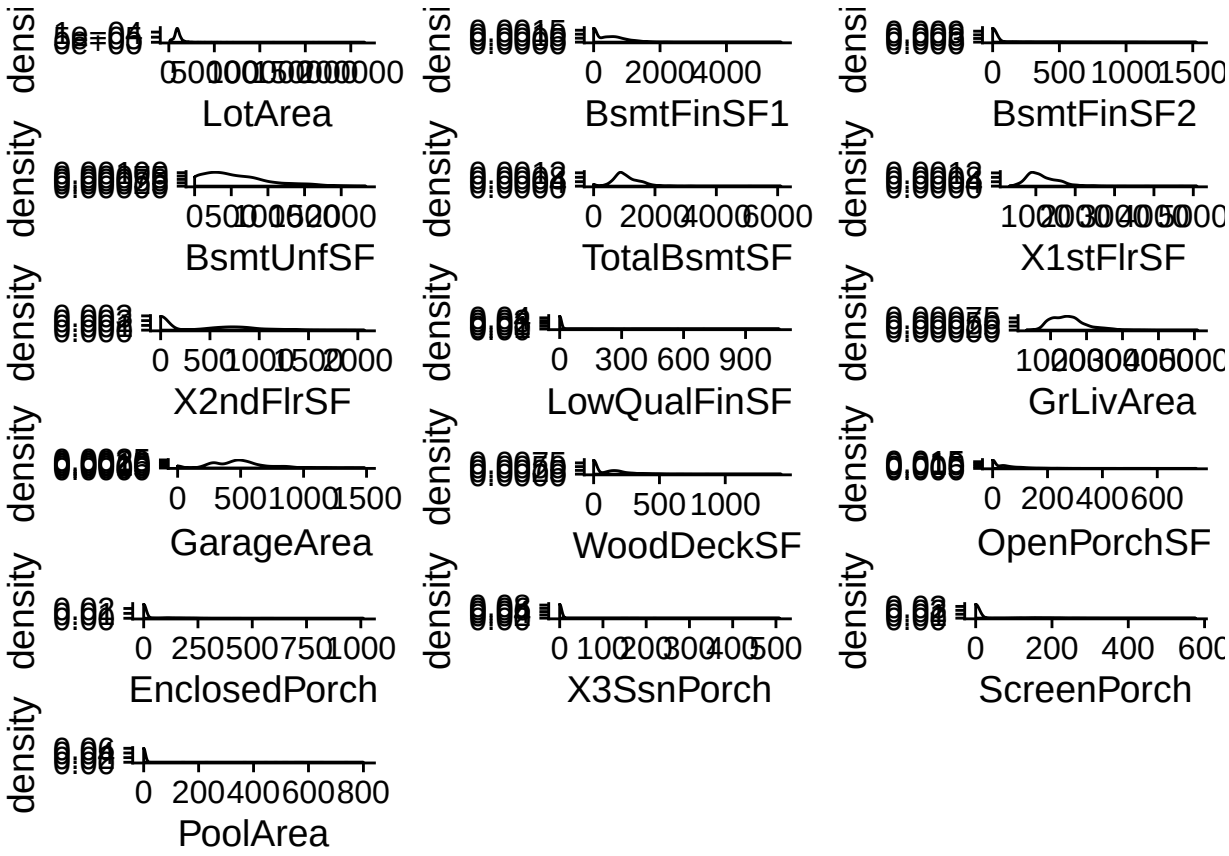
### Regularización de las variables continuas

Tal y como se ha mostrado en al análisis preliminar, la variable *SalePrice* contiene una distribución asimétrica. Por consiguiente, para evitar el efecto que los valores extremos puedan causar, se procede a aplicar logaritmos a los valores de la distribución.

```
full.set.transformed$SalePrice <- log(full.set.transformed$SalePrice)
```

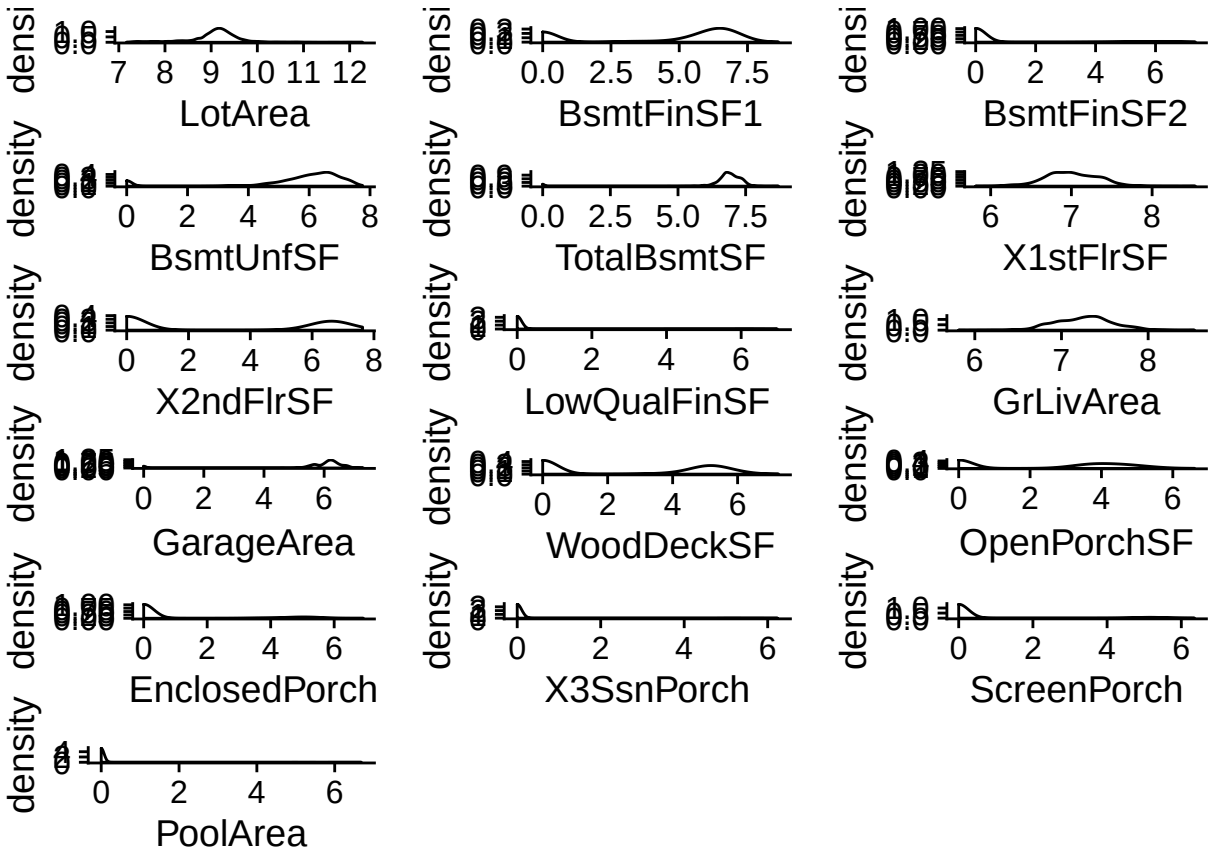
Así mismo, se procede a mostrar diagramas de densidad de cada una de las características que contienen datos númericos. De esta forma, se podrán observar la tendencias de cada una de las distribuciones.

```
continuous.features <- c(
  "LotArea", ## Lot size in square feet
  "BsmtFinSF1", ## Type 1 finished square feet
  "BsmtFinSF2", ## Type 2 finished square feet
  "BsmtUnfSF", ## Unfinished square feet of basement area
  "TotalBsmtSF", ## Total square feet of basement area
  "X1stFlrSF", ## First Floor square feet
  "X2ndFlrSF", ## Second floor square feet
  "LowQualFinSF", ## Low quality finished square feet (all floors)
  "GrLivArea", ## Above grade (ground) living area square feet
  "GarageArea", ## Size of garage in square feet
  "WoodDeckSF", ## Wood deck area in square feet
  "OpenPorchSF", ## Open porch area in square feet
  "EnclosedPorch", ## Enclosed porch area in square feet
  "X3SsnPorch", ## Three season porch area in square feet
  "ScreenPorch", ## Screen porch area in square feet
  "PoolArea" ## Pool area in square feet
)

plots <- lapply(continuous.features, function(feature) {
    if (is.numeric(full.set.transformed[, feature])) {
        ggplot2::ggplot(data = full.set.transformed,
        aes(x = full.set.transformed[, feature])) +
                    geom_density() +
                    xlab(feature)
    }
})

cowplot::plot_grid(plotlist = plots, ncol = 3)
```




Se aprecia una clara desviación en las distribuciones. Por consiguiente, aplicamos logaritmos a las caraterísticas de forma que en muchos casos obtenemos unas distribuciones más uniformes. Nótese que además se suma 1 a los datos, con el fin de evitar la operación *log(0)*.

```
full.set.transformed[, continuous.features] <- log(1 + full.set.transformed[,
    continuous.features])

plots <- lapply(continuous.features, function(feature) {
    if (is.numeric(full.set.transformed[, feature])) {
        ggplot2::ggplot(data = full.set.transformed, aes(x = full.set.transformed[,
            feature])) + geom_density() + xlab(feature)
    }
})

cowplot::plot_grid(plotlist = plots, ncol = 3)
```




**Otras transformaciones**

Con el fin de normalizar más los datos, se procede a aplicar un centrado y escalado de los conjunto de datos (extrayendo la variable clase).

```
set.seed(SEED)
full.set.preProcessed <- caret::preProcess(select(full.set.transformed,
    -SalePrice), method = c("center", "scale"))
full.set.transformed <- predict(full.set.preProcessed, full.set.transformed)
```

Algunos tipos de métodos analíticos que tienen problemas al tratar variables categóricas. Por lo tanto, se procede a la conversión de dicho tipo de variables a numéricas. Para ello, se asigna a cada categoría un entero, generado de forma incremental en cada característica.

```
for (i in 1:ncol(full.set.transformed)) {
    if (is.factor(full.set.transformed[, i])) {
        levels(full.set.transformed[, i]) <- c(1:length(levels(full.set.transformed[,
            i])))
        full.set.transformed[, i] <- as.numeric(full.set.transformed[,
            i])
    }
}
```

Así mismo, se ha observado que hay variables con una varianza muy cercana a 0. Aunque se utilizarán árboles de decisión para el modelado, este tipo de características suelen ser problemáticas y pueden hacer "quebrar" el modelo o volverlo inestable. Por consiguiente, se procede a indentificar dichas variables y eliminarlas.

```
near.zero.features.index <- caret::nearZeroVar(full.set.transformed)
full.set.transformed <- full.set.transformed[, -near.zero.features.index]
```

# Entrenamiento del modelo

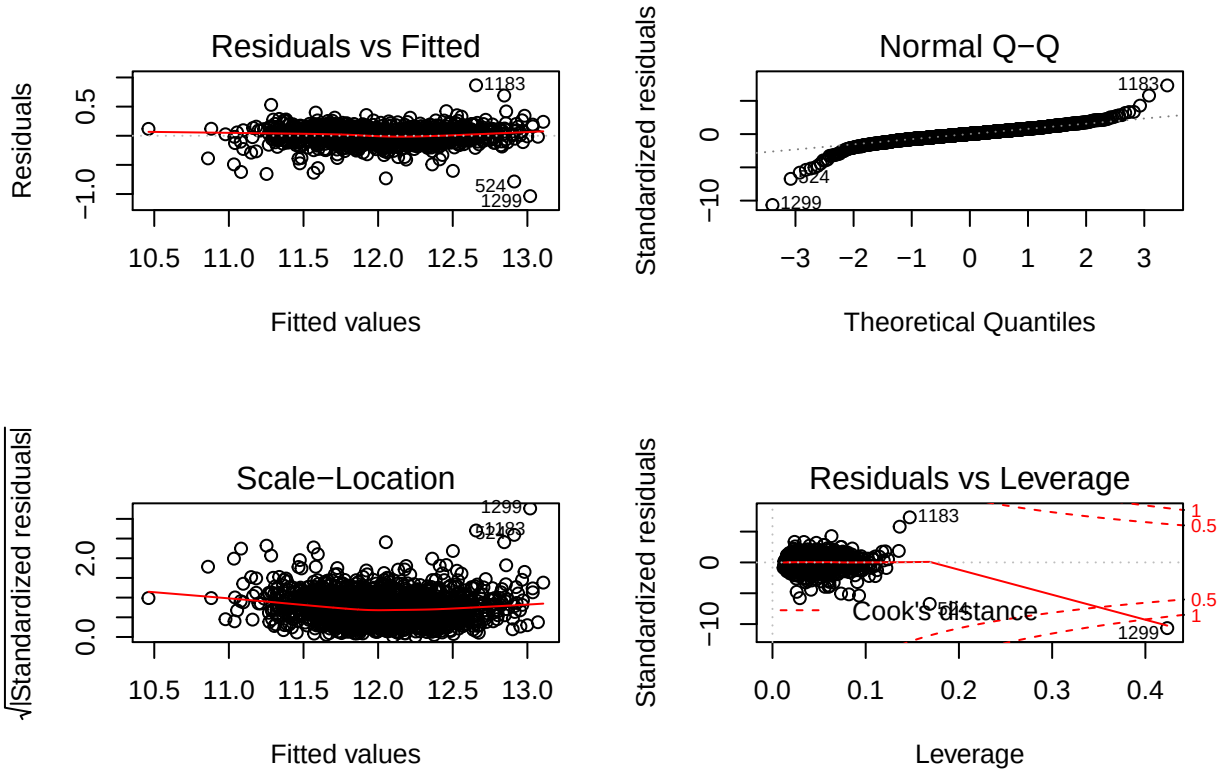
En primer lugar, se procede a volver a obtener los subconjuntos de entrenamiento y de test.

```
train.processed <- full.set.transformed[1:nrow(train.transformed),
    ]
test.processed <- full.set.transformed[(nrow(train.transformed) +
    1):nrow(full.set.transformed), ]
test.processed <- dplyr::select(test.processed, -SalePrice)
```

## Análisis exploratorio

De forma exploratoria, se procede a relizar una sencilla regresión linear y analizar la uniformación desprendida de su resultado. De esta forma obtenemos información sobre si nuestro conjunto de datos puede producir un buen modelo. Así, observando la primera gráfica se aprecia como los valores residuales se mantienen en torno a 0, especialmente en la zona de concentración de más instancias.

```
exploratory.lm = lm(SalePrice ~ ., data = train.processed)

par(mfrow = c(2, 2))
plot(exploratory.lm)
```



```
par(mfrow = c(1, 1))
```

También puede ser interesante concer cuales son las características con más impacto en la generación del modelo. A continuación, se muestra un ranking de las quince características más salientables. De forma destacada, la más ìmporante es *OverallQual*, dato que nos será relevante más adelante.

```
importance <- caret::varImp(exploratory.lm)
importance.sort <- sort(importance$Overall, decreasing = TRUE,
    index.return = TRUE)

data.frame(Feature = rownames(importance)[importance.sort$ix],
    Overall = importance[importance.sort$ix, ])[1:15, ]
```

```
##          Feature   Overall
## 1    OverallQual 14.751014
## 2      GrLivArea 12.580436
## 3    OverallCond 12.139590
## 4        LotArea  7.736403
## 5  SaleCondition  5.984287
## 6      YearBuilt  5.877734
## 7     GarageCars  5.377752
## 8     Fireplaces  4.580639
## 9     BsmtFinSF1  4.440173
## 10   KitchenQual  4.406557
## 11    CentralAir  4.081199
## 12  BsmtFullBath  3.445537
## 13      BsmtQual  3.378758
## 14     HeatingQC  3.211033
## 15   TotalBsmtSF  3.001786
```

## Entrenamiento parcial

Dado que nuestro conjunto de test no contiene información de la variable clase *SalePrice*, no se puede realizar una validación clásica del modelo. Es decir, se podría comparar el *RMSE* obtenido en la plataforma Kaggle y compraralo con el respectivo del conjunto de entrenamiento. Sin embargo, perderíamos mucha información en el proceso y no sería demasiado extensible.

Por lo tanto, se realiza una partición del conjunto de entrenamiento de forma que generemos dos nuevos subconjutos. Es decir, un nuevo subconjunto de entrenamiento (70% de las intancias del conjunto de entrenamiento original) y otro subconjunto de validación (formado por el 30% de intancias restantes).

```
set.seed(SEED)
train.processed.partition.index <- createDataPartition(train.processed$SalePrice,
    p = 0.7, list = FALSE)
train.processed.partition.train <- train.processed[train.processed.partition.index,
    ]
train.processed.partition.validation <- train.processed[-train.processed.partition.index,
    ]
```

Ahora se procede al entrenamiento del modelo. Se ha elegido un método de *Gradient boosting*, basado en árboles de decisión.

```
set.seed(SEED)
garbage <- capture.output(model.partial <- caret::train(SalePrice ~
    ., data = train.processed.partition.train, method = "gbm",
```

```
    trControl = caret::trainControl(method = "repeatedcv", number = 10,
        repeats = 1)))

model.partial
```

```
## Stochastic Gradient Boosting
##
## 1024 samples
##   57 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 1 times)
## Summary of sample sizes: 922, 923, 921, 920, 920, 922, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE       Rsquared   MAE
##   1                   50      0.1618867  0.8488163  0.11711994
##   1                  100      0.1397796  0.8769652  0.09990571
##   1                  150      0.1327752  0.8877363  0.09402571
##   2                   50      0.1424143  0.8745248  0.10108795
##   2                  100      0.1280579  0.8948544  0.09030664
##   2                  150      0.1242334  0.9008840  0.08697403
##   3                   50      0.1343270  0.8867105  0.09412257
##   3                  100      0.1243501  0.9007081  0.08590534
##   3                  150      0.1221374  0.9041962  0.08463566
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

### Validación del modelo

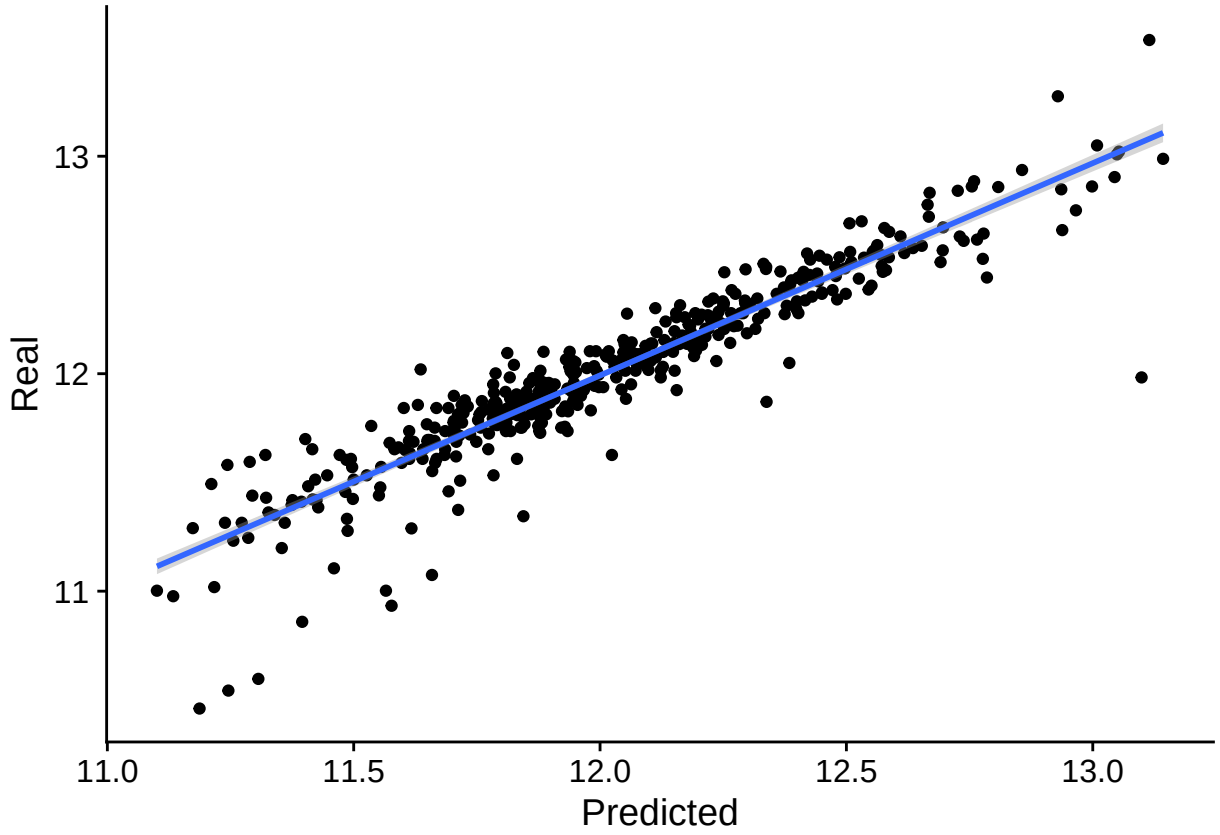
Para la validación del modelo, se crea la función *predictAndEvaluate*, de forma que se genere la respectiva predicción y se genere una gráfica comparativa entre los valores predichos y la respectiva *RSME*.

```
predictAndEvaluate <- function(model, validation.set) {
    validation.prediction <- stats::predict(model, dplyr::select(validation.set,
        -SalePrice))

    print(ggplot2::qplot(x = validation.prediction, y = validation.set$SalePrice,
        geom = c("point", "smooth"), method = "lm", xlab = "Predicted",
        ylab = "Real"))

    rmse(validation.set$SalePrice, validation.prediction)
}
```

Se procede a realizar la predicción del subconjunto de validación.

```
predictAndEvaluate(model.partial, train.processed.partition.validation)
```

```
## Warning: Ignoring unknown parameters: method
```




```
## [1] 0.150703
```

### Mejorando el modelo

Aunque no se han obtenido unos malos resultados, se quiere mejorar el entrenamiento del modelo. Para ello se ha creado al función *ensembleTrain*. Dicha función utiliza métodos del paquete *caretEnsemble*, de forma que permite combinar distintos modelos de *caret*. De esta forma, combinamos el *gbm* con otro tipo de árbol de decisión, un *xgbTree* o *eXtreme Gradient Boosting*. Así mismo, utlizamos la variable más relevante del dataset (*OverallQual*) para realizar un muestreo con *bootstrap*. Finalmente, juntamos los resultados basándonos en la métrica *RMSE*.

```
ensembleTrain <- function(train.set) {
    set.seed(SEED)

    trControl <- trainControl(method = "cv", number = 7, savePredictions = "final",
        index = createResample(train.set$OverallQual, 7), allowParallel = TRUE)

    garbage <- capture.output(modelList <- caretList(SalePrice ~
        ., data = train.set, trControl = trControl, metric = "RMSE",
        tuneList = list(gbm = caretModelSpec(method = "gbm",
            tuneGrid = expand.grid(n.trees = 700, interaction.depth = 5,
                shrinkage = 0.05, n.minobsinnode = 10)), xgbTree = caretModelSpec(method = "xgbTree",
            tuneGrid = expand.grid(nrounds = 2500, max_depth = 6,
                min_child_weight = 1.41, eta = 0.01, gamma = 0.0468,
                subsample = 0.769, colsample_bytree = 0.283)))))
```

```
    greedy_ensemble <- caretEnsemble(modelList, metric = "RMSE",
        trControl = trainControl(number = 25))

    return(greedy_ensemble)
}
```

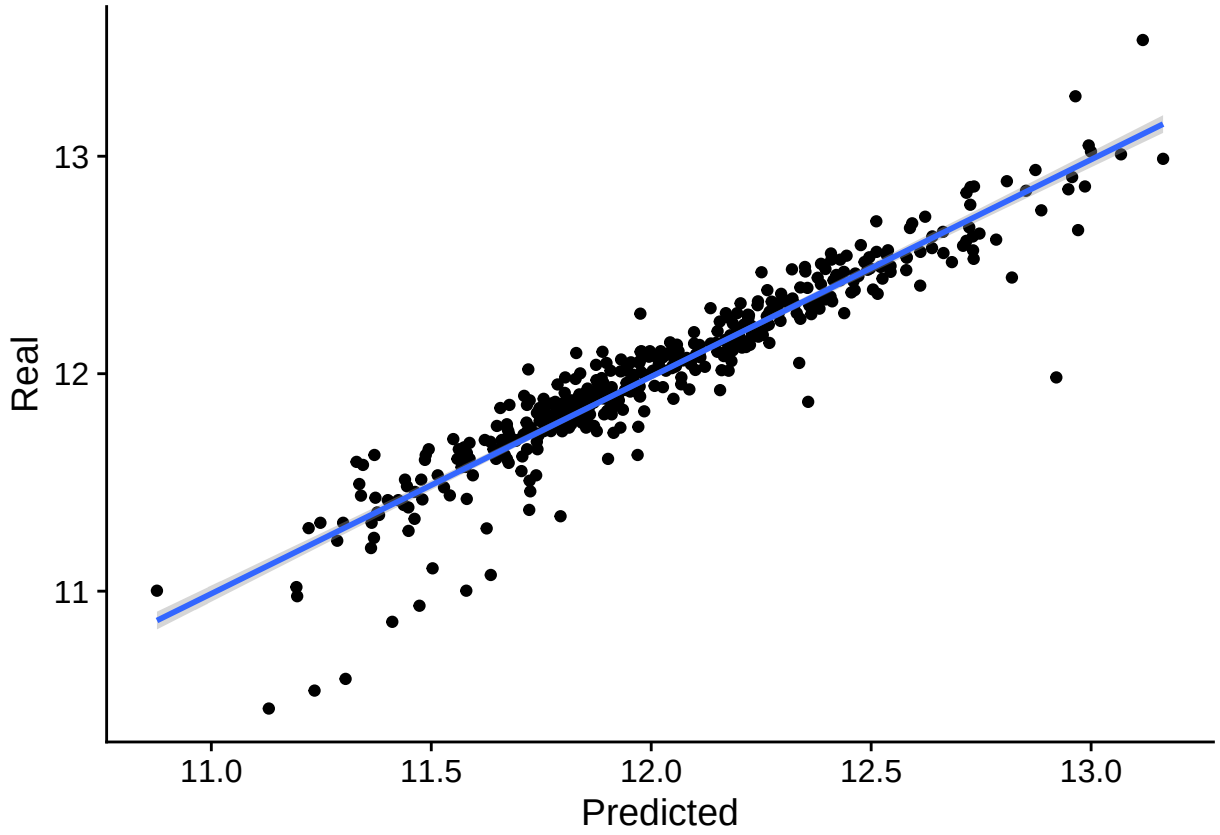
Se puede observar que tanto en el conjunto de entrenamiento como en el de validación, los valores han mejorado. Así mismo, la predicción parece correcta en la gráfica.

```
model.ensemble <- ensembleTrain(train.processed.partition.train)
model.ensemble
```

```
## A glm ensemble of 2 base models: gbm, xgbTree
##
## Ensemble results:
## Generalized Linear Model
##
## 2641 samples
##    2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 2641, 2641, 2641, 2641, 2641, 2641, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.1259608  0.8986272  0.08556768
```

```
predictAndEvaluate(model.ensemble, train.processed.partition.validation)
```

```
## Warning: Ignoring unknown parameters: method
```




```
## [1] 0.1400535
```

## Entrenamiento y predicción

Finalmente, se procede al entrenamiento de la totalidad del conjunto de entrenamiento y a su predicción. El valor de *RMSE* obtenido en la plataforma Kaggle es de 0.12332, por un 0.1257321 obtenido en el conjunto de entrenamiento. Ámbos son unos valores muy silimares, lo que nos confirma la ausencia de bias y varianza en el modelo contruido.

```
model.full <- ensembleTrain(train.processed)
model.full
```

```
## A glm ensemble of 2 base models: gbm, xgbTree
##
## Ensemble results:
## Generalized Linear Model
##
## 3736 samples
##    2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 3736, 3736, 3736, 3736, 3736, 3736, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
```

```
##   0.1257321  0.8999895  0.08361588
```

```
predictions <- predict(model.full, newdata = test.processed)
prediction.table <- data.frame(Id = test$Id, SalePrice = exp(predictions))
write.csv(prediction.table, "prediction.csv", row.names = FALSE)
```

# Referencias

A continuación se detallan los *kernels* de Kaggle utlizados tanto como inspiración como fuente de fragmentos de código:

- https://www.kaggle.com/wangyizhou30/svm-more5/code
- https://www.kaggle.com/philip198/starting-out-with-r-house-prices-prediction/code
- https://www.kaggle.com/bwboerman/r-data-table-glmnet-xgboost-with-caret/notebook
- https://www.kaggle.com/mariopasquato/support-vector-regression/code
- https://www.kaggle.com/dataygun/a-story-of-just-everything
- https://www.kaggle.com/notaapple/detailed-exploratory-data-analysis-using-r
- https://www.kaggle.com/jimthompson/regularized-linear-models-in-r