

# Fundamentos de la programación estadística y Data Mining en R

Dr. Germán Rosati (*Digital House - UNTREF - UNSAM*)

19 julio, 2017

## Regresión lineal

- Todos nos acordamos del modelo lineal: +

$$Y_i = \beta_0 + \beta_1 * X_i + \epsilon_i$$

- Los parámetros del modelo son muy fáciles de interpretar: +  $\beta_0$  es el intercepto +  $\beta_1$  es la pendiente de la variable  $X$ ; es decir el efecto medio en  $Y$  cuando  $X$  se incrementa en una unidad (y todo lo demás, se mantiene constante) +  $\epsilon_i$  es el error o residuo de estimación

## Regresión lineal

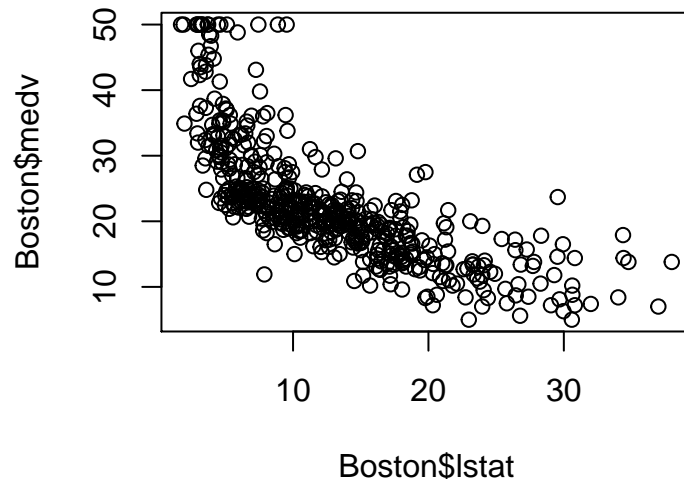
- En un modelo lineal buscamos “minimizar” una determinada métrica de error. En particular, buscamos hacer mínimo el error cuadrático medio (MSE): +

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

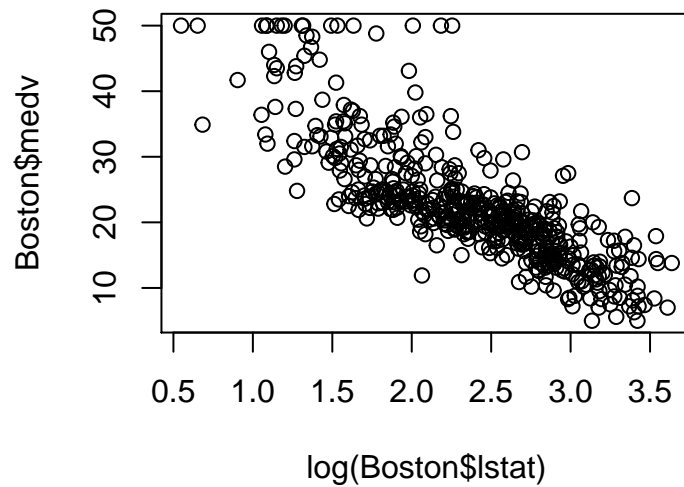
## Regresión lineal: implementación en R `lm()`

- Para implementar en R una regresión lineal simple usamos la función `lm()`
- **formula:** una expresión con la siguiente forma: `y~x`
- **data:** dataframe o datamatrix a utilizar
- **subset:** un vector que define un subconjunto de datos a usar en el modelo
- **weights:** vector que define pesos para la regresión (WLNS)

```
> library(MASS)
> data(Boston)
> plot(Boston$lstat, Boston$medv)
```



```
> plot(log(Boston$lstat), Boston$medv)
```



```
> model = lm(medv ~ log(lstat), data = Boston)
> model
```

```
##
## Call:
## lm(formula = medv ~ log(lstat), data = Boston)
##
## Coefficients:
## (Intercept)    log(lstat)
##      52.12      -12.48
```

- Si imprimimos el modelo... solamente nos da una información básica: el intercepto y el valor de la pendiente.

```
> summary(model)
##
## Call:
## lm(formula = medv ~ log(lstat), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.4599  -3.5006  -0.6686   2.1688  26.0129
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  52.1248    0.9652   54.00  <2e-16 ***
## log(lstat)  -12.4810    0.3946  -31.63  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.329 on 504 degrees of freedom
## Multiple R-squared:  0.6649, Adjusted R-squared:  0.6643
## F-statistic: 1000 on 1 and 504 DF, p-value: < 2.2e-16
```

- Ahora tenemos acceso a mucha más información:
- p-valores y errores estándar de los coeficientes. ¿Son significativos?
- $R^2$ : 66% de la variancia de la variable dependiente es explicada por el modelo

```
> names(model)
## [1] "coefficients" "residuals" "effects" "rank"
## [5] "fitted.values" "assign" "qr" "df.residual"
## [9] "xlevels" "call" "terms" "model"
```

- Usamos la función `names` para acceder a los objetos dentro del objeto `model`
- Luego, podemos ir usando los nombres para acceder a los diferentes elementos

```
> model$coefficients
## (Intercept) log(lstat)
## 52.12476 -12.48097
```

- Algunas funciones útiles:

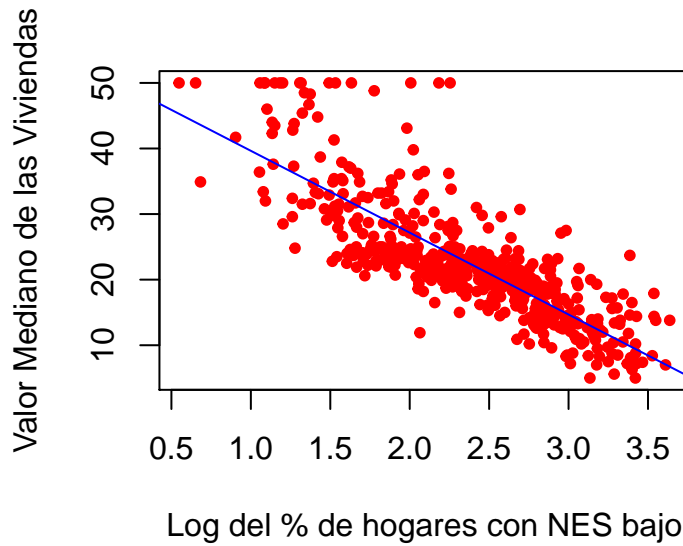
```
> coef(model)
## (Intercept) log(lstat)
## 52.12476 -12.48097
> confint(model, level = 0.95)
## 2.5 % 97.5 %
## (Intercept) 50.22846 54.02105
## log(lstat) -13.25631 -11.70564
```

- Obtenemos intervalos de confianza de los parámetros del modelo.
- Veamos la función `predict()`

```
> predict(model, data.frame(lstat = c(5, 10, 15), interval = "prediction"))
## 1 2 3
## 32.03741 23.38626 18.32566
```

- Es decir, que para si la variable independiente `lstat` presentara los valores 5, 10 y 15, la variable dependiente `medv` presentaría esos valores (en la media).
- Grafiquemos, ahora, todo.

```
plot(log(Boston$lstat), Boston$medv,
     xlab = "Log del % de hogares con NES bajo",
     ylab = "Valor Mediano de las Viviendas",
     col = "red",
     pch = 20)
abline(model, col="blue")
```



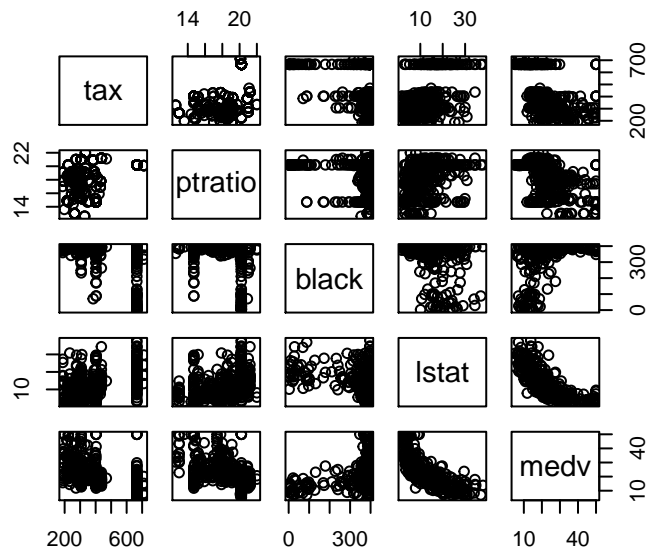
### Regresión lineal múltiple: implementación en R `lm()`

- Generemos un modelo, ahora, que contenga todas las variables del dataset.
- Veamos, primero, la correlación entre varias variables:

```
> cor(Boston[, 10:14])
##           tax      ptratio      black      lstat      medv
## tax      1.0000000  0.4608530 -0.4418080  0.5439934 -0.4685359
## ptratio  0.4608530  1.0000000 -0.1773833  0.3740443 -0.5077867
## black   -0.4418080 -0.1773833  1.0000000 -0.3660869  0.3334608
## lstat    0.5439934  0.3740443 -0.3660869  1.0000000 -0.7376627
## medv    -0.4685359 -0.5077867  0.3334608 -0.7376627  1.0000000
```

- La función `cor` tiene varios argumentos
- `x`, `y`: las variables (si todas son cuantitativas podemos pasar todo el dataframe)
- `method`: qué coeficiente(s) se va(n) a usar... ¿Pearson, Spearman o Kendall?
- Mucho mejor es verlo en una matriz de gráficos...

```
pairs(Boston[,10:14])
```



- Implementemos un modelo con todas las variables

```
> model <- lm(medv ~ ., data = Boston)
> summary(model)
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00 -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116 < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01 -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

- Pero momento... habíamos precisado en el primer modelo que lstat entraba en forma logarítmica...

```

> model <- lm(medv ~ . - lstat + log(lstat), data = Boston)
> summary(model)
##
## Call:
## lm(formula = medv ~ . - lstat + log(lstat), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.9478  -2.6350  -0.2669   1.8253  24.7475
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  55.702945   4.873358  11.430 < 2e-16 ***
## crim        -0.127911   0.029118  -4.393 1.37e-05 ***
## zn           0.024383   0.012294   1.983 0.047876 *
## indus        0.016095   0.054910   0.293 0.769553
## chas         2.136842   0.772166   2.767 0.005865 **
## nox        -16.192209   3.418741  -4.736 2.85e-06 ***
## rm           2.415005   0.391483   6.169 1.44e-09 ***
## age          0.026899   0.012000   2.242 0.025430 *
## dis         -1.235606   0.179367  -6.889 1.73e-11 ***
## rad           0.301501   0.059315   5.083 5.29e-07 ***
## tax         -0.011377   0.003363  -3.383 0.000774 ***
## ptratio     -0.861362   0.117277  -7.345 8.62e-13 ***
## black        0.007670   0.002402   3.193 0.001498 **
## log(lstat)  -9.167803   0.571984 -16.028 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.244 on 492 degrees of freedom
## Multiple R-squared:  0.7925, Adjusted R-squared:  0.7871
## F-statistic: 144.6 on 13 and 492 DF,  p-value: < 2.2e-16

```

## Regresión lineal múltiple: implementación en R `vif()`

- Vamos a analizar la multicolinealidad entre los predictores. Usaremos la función `vif()` del paquete `car`

```

> library(car)
> vif(model)
##      crim      zn      indus      chas      nox      rm
##  1.758684  2.304733  3.978377  1.078410  4.399940  2.121180
##      age      dis      rad      tax      ptratio      black
##  3.198699  3.999433  7.478269  9.006844  1.807326  1.348314
## log(lstat)
##  3.311881

```

- Pareciera que `tax` y `rad` están muy correlacionadas con al menos un predictor. Entonces, podríamos reestimar el modelo eliminándolas.

```
> model <- lm(medv ~ . - lstat + log(lstat) - tax - rad, data = Boston)
```

## Regresión lineal múltiple: interacciones

- Supongamos que quisiéramos agregar alguna interacción podemos hacerlo con la siguiente sintaxis `log(lstat)*age` que agrega tanto los términos de interacción como los efectos de cada variable por separado.
- Si quisiéramos introducir solamente la interacción deberíamos usar `log(lstat)*age`

```
> mod <- lm(medv ~ log(lstat) * age, data = Boston)
> model <- lm(medv ~ log(lstat) * age, data = Boston)
> mod
##
## Call:
## lm(formula = medv ~ log(lstat) * age, data = Boston)
##
## Coefficients:
##      (Intercept)      log(lstat)          age  log(lstat):age
##          45.65479         -11.01171          0.15052         -0.04244
> model
##
## Call:
## lm(formula = medv ~ log(lstat) * age, data = Boston)
##
## Coefficients:
##      (Intercept)      log(lstat)          age  log(lstat):age
##          45.65479         -11.01171          0.15052         -0.04244
```