Data Science con R

Instituto de Estadística PUCV - Magister en Estadística

Dae-Jin Lee < dlee@bcamath.org >

```
install.packages("visreg")
install.packages("car")
install.packages("leaps")
install.packages("InformationValue")
install.packages("ROCR")
```

Modelo Lineal General

Modelos lineales en R

Regresión lineal simple

- La regresión es un método estadístico utilizado para predecir el valor de una variable de respuesta basada en los valores de un conjunto de variables explicativas.
- Cualquier modelo estadístico intenta aproximar la variable de respuesta o variable dependiente y como una función matemática de las variables explicativas o regresores X (también llamadas covariables o variables independientes).
- La forma más sencilla y más común es la **regresión lineal**

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_p + \epsilon,$$

donde β_i con $i = 0, 1, 2$ son parametros *desconocidos*. β_0 se llama el intercepto. Por lo tanto, el problema se reduce a la estimación de cuatro valores. ϵ es un término de error aleatorio que se asume Normal de media 0 y varianza σ^2. El modelo lineal asume que los errores son además homocedásticos (varianza constante).

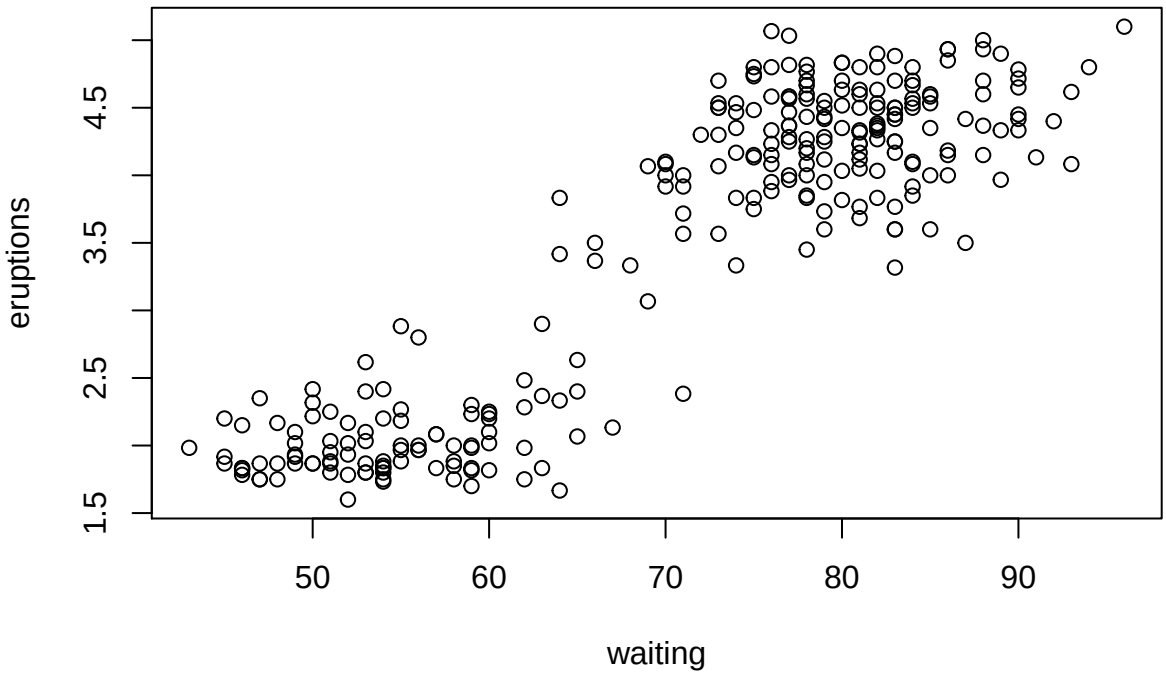
- Un modelo lineal simple con una sola variable explicativa se define como:

$$\hat{y} = \beta_0 + \beta_1 x$$

donde $\hat{y}$ son los valores ajustados para β_0 (intercepto) y β_1 (pendiente). Entonces para un x_i dado obtenemos un $\hat{y}_i$ que se aproxima a y_i

Por ejemplo, en el conjunto de datos `faithful`, contiene datos de ejemplo de dos variables aleatorias denominadas `waiting` y `eruptions`. La variable `waiting` indica el tiempo de espera hasta las próximas erupciones, y`eruptions` denota la duración.

```
plot(eruptions~waiting,data=faithful)
```



El modelo lineal viene dado por:

$$Eruptions = \beta_0 + \beta_1 * Waiting + \epsilon$$

Si seleccionamos los parámetros β_0 y β_1 en el modelo de regresión lineal simple para minimizar la suma de cuadrados del término de error ϵ. Supongamos que para el conjunto de datos `faithful`, pretendemos estimar la siguiente duración de la erupción si el tiempo de espera desde la última erupción ha sido de 80 minutos. Aplicamos la función `lm` a una fórmula que describe la variable erupciones por la variable waiting, y guardamos el modelo de regresión lineal en una nueva variable `eruption.lm`.

```
data("faithful")
eruption.lm <- lm(eruptions~waiting,data=faithful)
```

Extraemos los parámetros de la ecuación de regresión estimada con la función de coeficientes.

```
coeffs <- coefficients(eruption.lm); coeffs
```

```
## (Intercept)     waiting
## -1.87401599  0.07562795
```

Ahora ajustamos la duración de la erupción usando la ecuación de regresión estimada.

```
waiting = 80           # the waiting time
duration = coeffs[1] + coeffs[2]*waiting
duration
```

```
## (Intercept)
##     4.17622
```

En base al modelo de regresión lineal simple, si el tiempo de espera desde la última erupción ha sido de 80 minutos, esperamos que los próximos minutos de duración 4.1762198.

Podemos incluir el valor de `waiting=80` en `newdata` como un `data.frame`

```
newdata = data.frame(waiting=80) # wrap the parameter
```

Y aplicar la función `predict` a modelo `eruption.lm` con `newdata`.

```
predict(eruption.lm, newdata)    # apply predict
```

```
##       1
## 4.17622
```

Definir modelos en R

Para completar una regresión lineal usando R es necesario primero entender la sintaxis para definir modelos.

Syntax	Model	Comments
`y ~ x`	$y = \beta_0 + \beta_1 x$	Linea recta con intercepto
`y ~ -1 + x`	$y = \beta_1 x$	Linea recta sin intercepto, i.e. la recta pasa por (0,0)
`y ~ x + I(x^2)`	$y = \beta_0 + \beta_1 x + \beta_2 x^2$	Modelo polinomial; `I()` permite incluir símbolos matemáticos
`y ~ x + z`	$y = \beta_0 + \beta_1 x + \beta_2 z$	Model de regresión múltiple
`y ~ x:z`	$y = \beta_0 + \beta_1 xz$	Modelo con interacción entre x y z
`y ~ x*z`	$y = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz$	Equivale a `y~x+z+x:z`

La función `lm()` permite calcular internamente el modelo lineal de regresión.

Con el comando `names()` podemos ver los componentes de un objeto de R

```
names(eruption.lm)
```

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

Por ejemplo:

- Valores ajustados (o predichos) ($\hat{y}$):

```
eruption.lm$fitted.values
```

- Residuos ($y - \hat{y}$)

```
eruption.lm$residuals
```

Podemos estimar σ como $\sigma = \frac{(y_i - \hat{y}_i)^2}{n-p}$ en R

```
eruption.lm.sum <- summary(eruption.lm)
names(eruption.lm.sum)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
sqrt(deviance(eruption.lm)/df.residual(eruption.lm))
```

```
## [1] 0.4965129
```

```
# is obtained directly as
eruption.lm.sum$sigma
```

```
## [1] 0.4965129
```

Coeficiente de determinación

El **coeficiente de determinación** de un modelo de regresión lineal es el cociente de las varianzas de los valores ajustados y los valores observados de la variable dependiente. Si denotamos y_i como los valores observados de la variable dependiente, $\bar{y}$ como su media, y $\bar{y}_i$ como el valor ajustado, entonces el coeficiente de determinación es:

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{(y_i - \bar{y})^2}$$

```
summary(eruption.lm)$r.squared
```

```
## [1] 0.8114608
```

Más opciones:

- `fitted.values()` o `fitted()` valores ajustados
- `predict()`: valores predichos $\hat{y}_*$ para valores de x_*
- `confint()`: intervalos de confianza para los parámetros del modelo
- `resid()`: residuos del modelo
- `anova()`: Tabla de analisis de la varianza para los residuos
- `deviance()`: devianza del modelo ajustado, en el caso de la regresión lineal $\sum_i^n (\hat{y}_i - y_i)^2$

Ver libro de Faraway (2002) book (Chapters 1-7) aquí

Prueba de significatividad para la regresión lineal

Supongamos que el término de error en el modelo de regresión lineal es independiente de x, y se distribuye normalmente, con media cero y varianza constante. Podemos decidir si existe alguna relación significativa entre x e y probando la hipótesis nula de que $\beta_1 = 0$.

El resultado del test es el estadístico F

```
summary(eruption.lm)
```

```
##
## Call:
## lm(formula = eruptions ~ waiting, data = faithful)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.29917 -0.37689  0.03508  0.34909  1.19329
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.874016   0.160143  -11.70   <2e-16 ***
## waiting      0.075628   0.002219   34.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4965 on 270 degrees of freedom
## Multiple R-squared:  0.8115, Adjusted R-squared:  0.8108
## F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16
```

Intervalo de confianza para la regresión lineal

Supongamos que el término de error ϵ en el modelo de regresión lineal es independiente de x, y se distribuye normalmente, con media cero y varianza constante. Para un valor dado de x, la estimación de intervalo para la media de la variable dependiente, $\bar{y}$, se llama intervalo de confianza.

Un intervalo de confianza del 95% de la duración media de la erupción para el tiempo de espera de 80 minutos está dado por

```
predict(eruption.lm, newdata, interval="confidence")
```

```
##       fit      lwr      upr
## 1 4.17622 4.104848 4.247592
```

El intervalo de confianza del 95% de la duración media de la erupción para el tiempo de espera de 80 minutos está entre 4.1048 y 4.2476 minutos.

Intervalo de predicción para la regresión lineal

Para un valor dado de x, la estimación de intervalo de la variable dependiente y se denomina intervalo de predicción.

```
predict(eruption.lm, newdata, interval="predict")
```

```
##       fit      lwr      upr
## 1 4.17622 3.196089 5.156351
```

El intervalo de predicción del 95% de la duración de la erupción para el tiempo de espera de 80 minutos está entre 3.1961 y 5.1564 minutos.

NOTA:

- Un **intervalo de predicción** es un intervalo asociado con una variable aleatoria aún no observada (predicción).
- Un **intervalo de confianza** es un intervalo asociado a un parámetro y es un concepto de frecuentista.

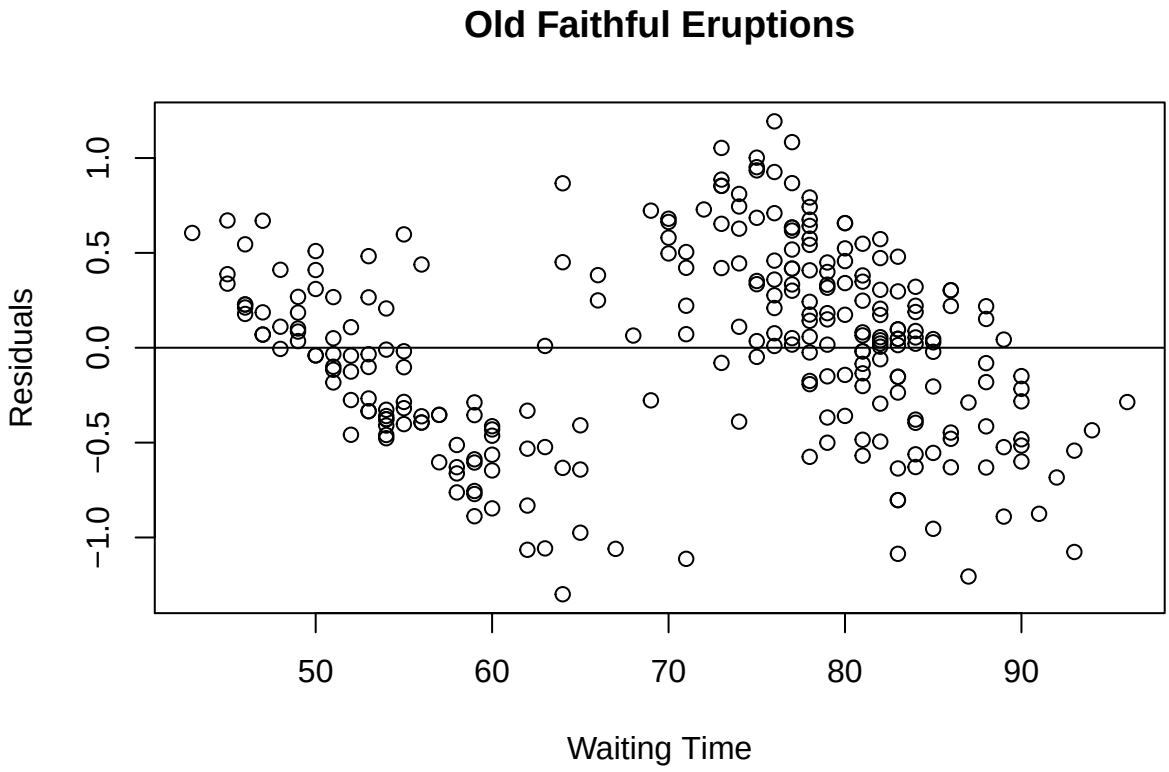
Gráfico de residuos

Los residuos del modelo de regresión lineal simple son la diferencia entre los datos observados de la variable dependiente y y los valores ajustados $\hat{y}$.

$$Residuos = y - \hat{y}$$

```
eruption.res = resid(eruption.lm)
```

```
plot(faithful$waiting,
    eruption.res,
    ylab="Residuals", xlab="Waiting Time",
    main="Old Faithful Eruptions")
abline(0, 0)
```



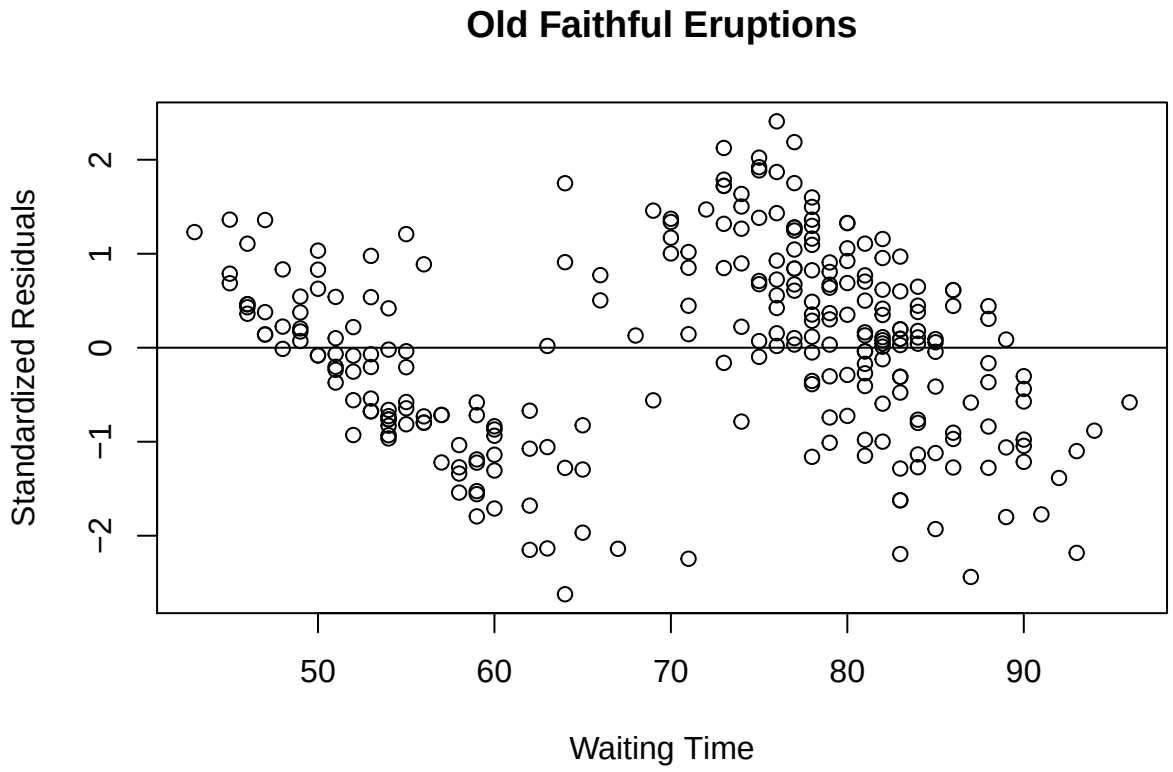
Residuo estandarizado

El residuo estandarizado es el residuo dividido por su desviación estándar.

$$\text{Standardized residual}_i = \frac{Residual_i}{SD.of.Residual_i}$$

```
eruption.stdres <- rstandard(eruption.lm)
```

```
plot(faithful$waiting, eruption.stdres,
      ylab="Standardized Residuals",
      xlab="Waiting Time",
      main="Old Faithful Eruptions")
abline(0, 0)
```



Como el p-valor es mucho menor que 0.05, rechazamos la hipótesis nula de que $\beta_1 = 0$. Por lo tanto, existe una relación significativa entre las variables en el modelo de regresión lineal del conjunto de datos `faithful`.

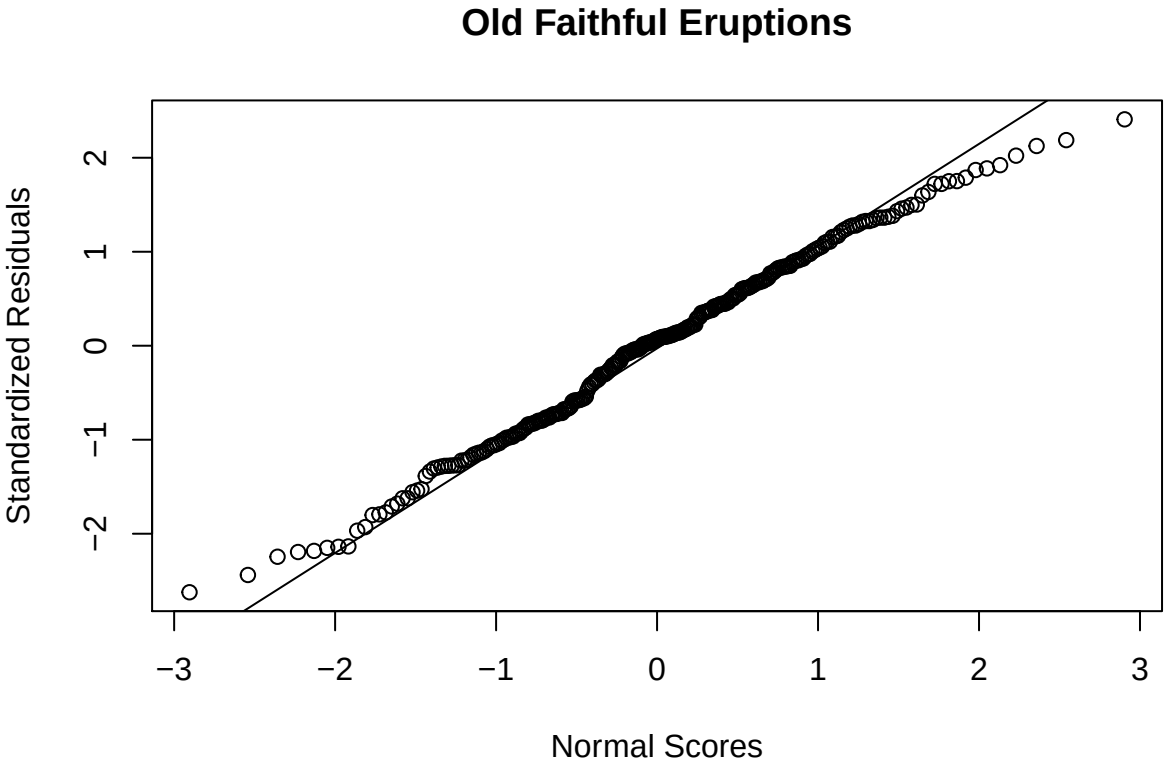
Gráfico de normalidad de los residuos

El gráfico de probabilidad normal es una herramienta gráfica para comparar un conjunto de datos con la distribución normal. Podemos usarlo con el residuo estandarizado del modelo de regresión lineal y ver si el término de error ϵ es realmente distribuido normalmente.

```
eruption.lm = lm(eruptions ~ waiting, data=faithful)
eruption.stdres = rstandard(eruption.lm)
```

Ahora creamos el gráfico de probabilidad normal con la función `qqnorm` y añadimos`qqline` para una comparación posterior.

```
qqnorm(eruption.stdres,
    ylab="Standardized Residuals",
    xlab="Normal Scores",
    main="Old Faithful Eruptions")
qqline(eruption.stdres)
```



Regresión lineal múltiple

Un modelo de regresión lineal múltiple describe una variable dependiente y por variables independientes $x_1, x_2, ..., x_p$ $(p > 1)$ se expresa mediante la ecuación:

$$y = \beta_0 + \sum_{k}^{p} x_k \beta_k + \epsilon$$

donde β_0 y β_k $(k = 1, 2, ..., p)$ son los parámetros y ϵ el término de error.

Ejemplo: `data(stackloss)`

Datos de funcionamiento de una planta de oxidación de amoniaco a ácido nítrico. Los datos se han obtenido a partir de 21 días de operación de una planta de oxidación de amoniaco (NH3) a ácido nítrico (HNO3). Los óxidos nítricos producidos se absorben en una torre de absorción de contracorriente.

El flujo de aire (`Air.Flow`) representa la tasa de operación de la planta. `Water.Temp` es la temperatura del agua de refrigeración que circula a través de los serpentines de la torre de absorción. `Acid.Conc.` es la concentración del ácido que circula, menos 50, por 10: es decir, 89 corresponde al 58,9 por ciento de ácido.

`stack.loss` es la pérdida por acumulación (la variable dependiente) es 10 veces el porcentaje del amoníaco entrante a la planta que escapa de la columna de absorción no absorbida; es decir, una medida (inversa) de la eficiencia general de la planta.

La regresión multiple sería:

$$stack.loss = \beta_0 + \beta_1 * Air.Flow + \beta_2 * Water.Temp + \beta_3 * Acid.Conc + \epsilon$$

```r
data("stackloss")
?stackloss
head(stackloss)
```

```
##   Air.Flow Water.Temp Acid.Conc. stack.loss
## 1       80         27         89         42
## 2       80         27         88         37
## 3       75         25         90         37
## 4       62         24         87         28
## 5       62         22         87         18
## 6       62         23         87         18
```

El modelo de regresión multiple en `R` vendría dado por:

```r
stackloss.lm = lm(stack.loss ~ Air.Flow + Water.Temp + Acid.Conc., data=stackloss)
stackloss.lm
```

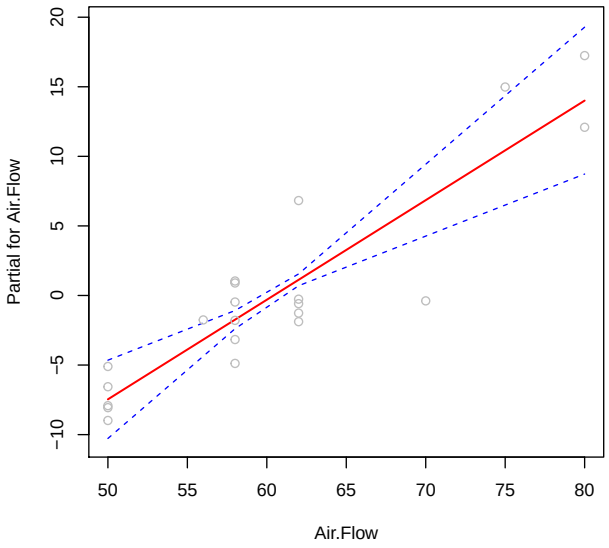
```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Coefficients:
## (Intercept)     Air.Flow   Water.Temp   Acid.Conc.
##    -39.9197       0.7156       1.2953      -0.1521
```

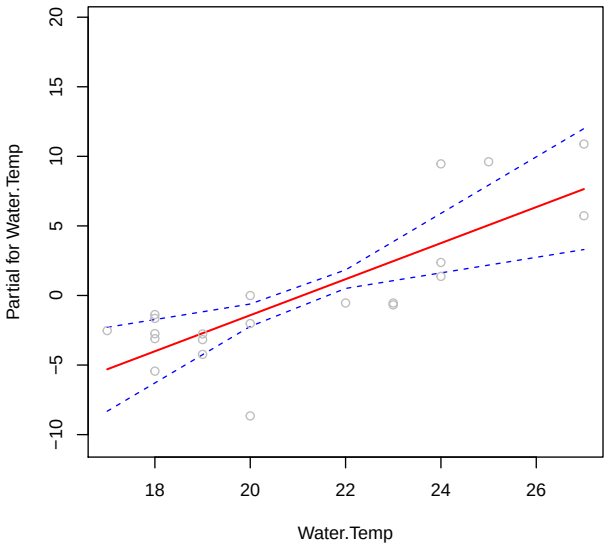
```r
summary(stackloss.lm)
```

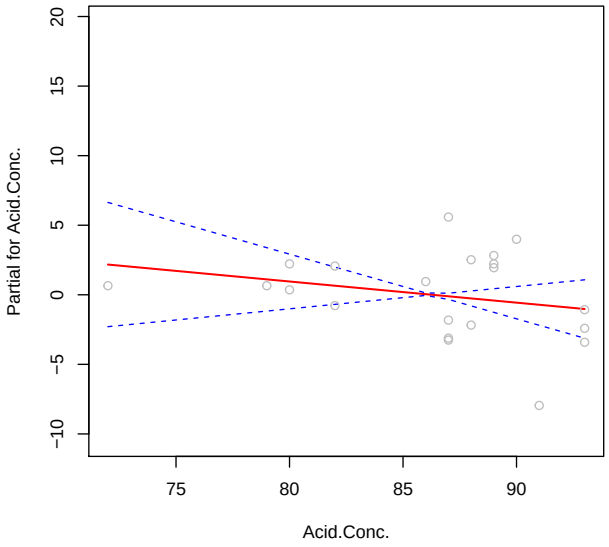
```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

La función `termplot` permite representar los términos de la regresión frente a las variables predictoras:

```r
?termplot
par(mfrow=c(2,2))
termplot(stackloss.lm, partial.resid = TRUE, se=TRUE,col.se = "blue")
```







¿Cuál es la pérdida de la chimenea si el flujo de aire es 72, la temperatura del agua es 20 y la concentración de ácido es 85?

Crear un nuevo `data.frame`:

```
newdata <- data.frame(Air.Flow=72,Water.Temp=20,Acid.Conc.=85)
```

Con `predict`

```
predict(stackloss.lm, newdata)
```

```
##        1
## 24.58173
```

Basado en el modelo de regresión lineal múltiple y los parámetros dados, la pérdida prevista es 24.5817284.

Para obtener el coeficiente de determinación múltiple

```r
summary(stackloss.lm)$r.squared
```

```
## [1] 0.9135769
```

Coeficiente de determinación ajustado

El coeficiente de determinación ajustado de un modelo de regresión lineal múltiple se define en términos del coeficiente de determinación como sigue, donde n es el número de observaciones en el conjunto de datos y p es el número de variables independientes.

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

```r
summary(stackloss.lm)$adj.r.squared
```

```
## [1] 0.8983258
```

Pruebas de significación e intervalos de confianza / predicción

```r
summary(stackloss.lm)
```

```
##
## Call:
## lm(formula = stack.loss ~ Air.Flow + Water.Temp + Acid.Conc.,
##     data = stackloss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2377 -1.7117 -0.4551  2.3614  5.6978
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -39.9197    11.8960  -3.356  0.00375 **
## Air.Flow      0.7156     0.1349   5.307  5.8e-05 ***
## Water.Temp    1.2953     0.3680   3.520  0.00263 **
## Acid.Conc.   -0.1521     0.1563  -0.973  0.34405
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.243 on 17 degrees of freedom
## Multiple R-squared:  0.9136, Adjusted R-squared:  0.8983
## F-statistic:  59.9 on 3 and 17 DF,  p-value: 3.016e-09
```

Como los p-valores de `Air.Flow` y`Water.Temp` son inferiores a 0,05, ambos son estadísticamente significativos en el modelo de regresión lineal múltiple de `stackloss`.

El intervalo de confianza al 95% de `stackloss` es

```r
predict(stackloss.lm, newdata, interval="confidence")
```

```
##        fit      lwr    upr
## 1 24.58173 20.21846 28.945
```

Y el invervalo de predicción al 95%

```
predict(stackloss.lm, newdata, interval="prediction")
```

```
##        fit     lwr      upr
## 1 24.58173 16.4661 32.69736
```

Regresión lineal con variables factor

Supongamos el conjunto de datos `mtcars`.

```
data(mtcars)
t.test(mpg ~ am, data=mtcars)
```

```
##
##  Welch Two Sample t-test
##
## data:  mpg by am
## t = -3.7671, df = 18.332, p-value = 0.001374
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -11.280194  -3.209684
## sample estimates:
## mean in group 0 mean in group 1
##        17.14737        24.39231
```

Los resultados de las pruebas estadísticas se centran en `mpg` y`am` solamente, sin controlar las influencias de otras variables.

```
fit0 <- lm(mpg ~ factor(am), data = mtcars)
summary(fit0)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3923 -3.0923 -0.2974  3.2439  9.5077
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   17.147      1.125  15.247 1.13e-15 ***
## factor(am)1    7.245      1.764   4.106 0.000285 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.902 on 30 degrees of freedom
## Multiple R-squared:  0.3598, Adjusted R-squared:  0.3385
## F-statistic: 16.86 on 1 and 30 DF,  p-value: 0.000285
```

Si aplicamos una regresión multiple para controlar ciertas variables disponibles de diseño y rendimiento, el impacto marginal de los automóviles de transmisión automática o manual no resulta significativo. Las variables de confusión incluyen desplazamiento (`disp`), relación del eje trasero (`drat`) y peso del coche (`wt`). Supongamos el peso del coche (`wt`) por ejemplo. La regresión sugiere que, manteniendo constantes otras variables (ceteris paribus), los automóviles traídos por tracción consumen -0.024 galones más de gas por milla

y los resultados ya no son estadísticamente significativos. Un análisis similar se puede observar para las otras dos variables: `drat` y `wt`.

```
fit1 <- lm(mpg ~ factor(am) + wt, data = mtcars)
summary(fit1)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5295 -2.3619 -0.1317  1.4025  6.8782
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 37.32155    3.05464  12.218 5.84e-13 ***
## factor(am)1 -0.02362    1.54565  -0.015    0.988
## wt          -5.35281    0.78824  -6.791 1.87e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.098 on 29 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7358
## F-statistic: 44.17 on 2 and 29 DF,  p-value: 1.579e-09
```

```
fit2 <- lm(mpg ~ factor(am) + drat, data = mtcars)
summary(fit2)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + drat, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5802 -2.5206 -0.5153  2.4419  8.5198
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.950      7.073  -0.276   0.7848
## factor(am)1    2.807      2.282   1.230   0.2286
## drat           5.811      2.130   2.728   0.0107 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.448 on 29 degrees of freedom
## Multiple R-squared:  0.4906, Adjusted R-squared:  0.4554
## F-statistic: 13.96 on 2 and 29 DF,  p-value: 5.659e-05
```

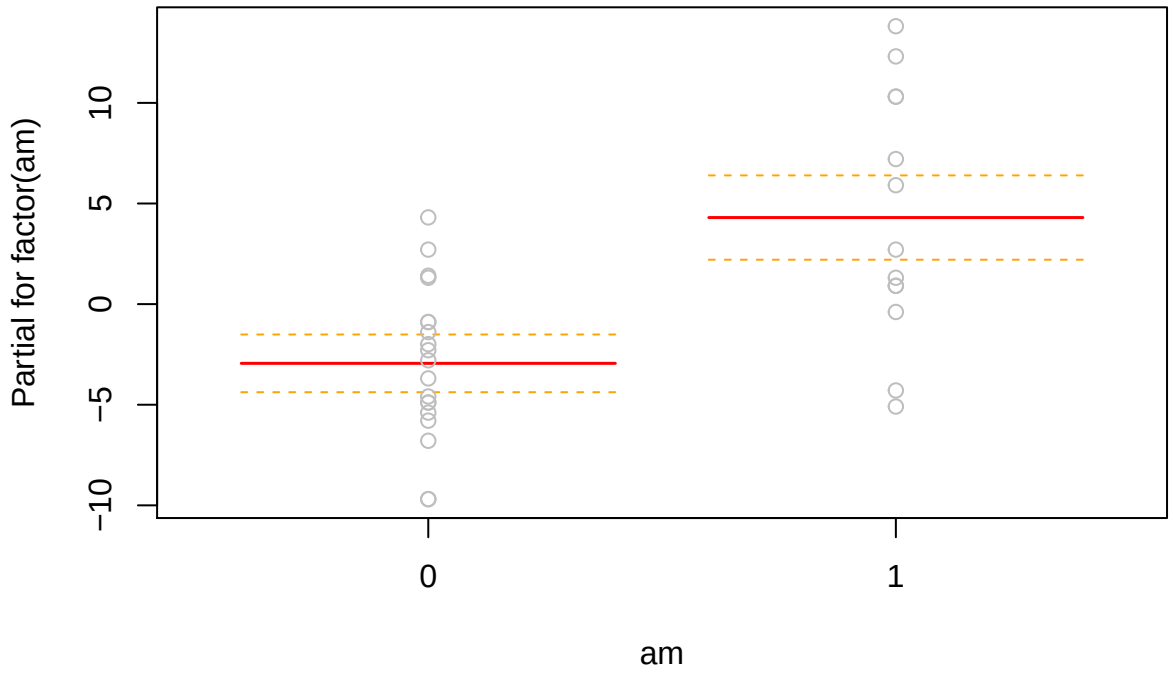
```
fit3 <- lm(mpg ~ factor(am) + disp, data = mtcars)
summary(fit3)
```

```
##
## Call:
## lm(formula = mpg ~ factor(am) + disp, data = mtcars)
##
```

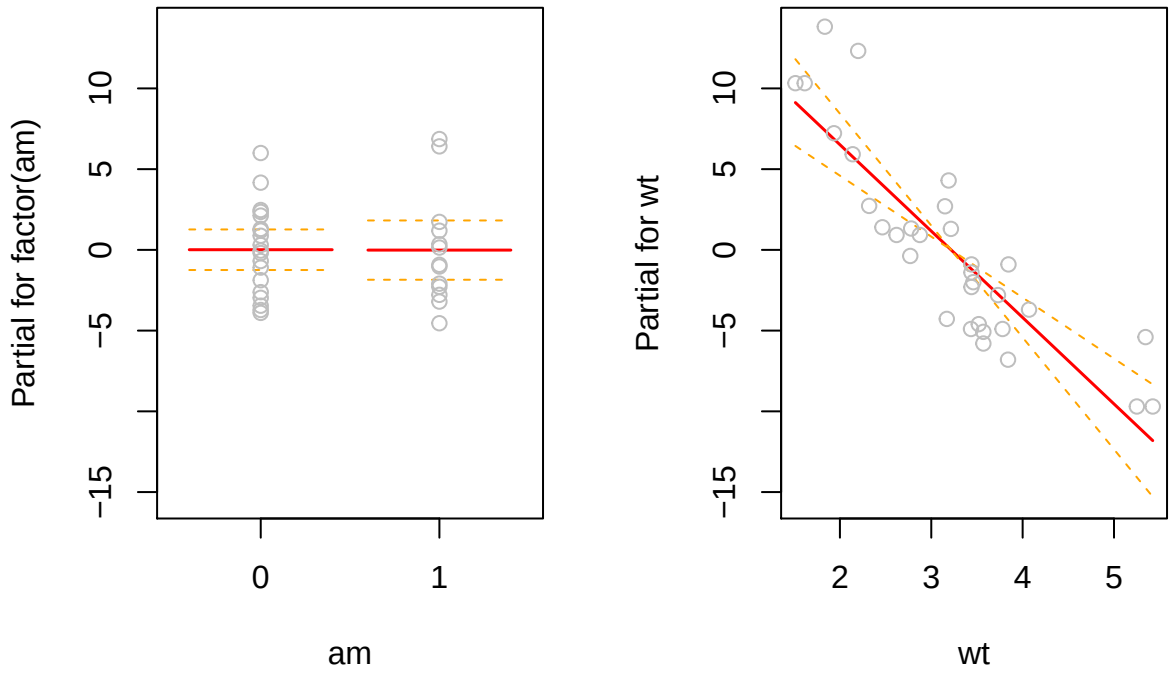
```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6382 -2.4751 -0.5631  2.2333  6.8386
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.848081   1.834071  15.184 2.45e-15 ***
## factor(am)1  1.833458   1.436100   1.277    0.212
## disp        -0.036851   0.005782  -6.373 5.75e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.218 on 29 degrees of freedom
## Multiple R-squared:  0.7333, Adjusted R-squared:  0.7149
## F-statistic: 39.87 on 2 and 29 DF,  p-value: 4.749e-09
```

Con `termplot`

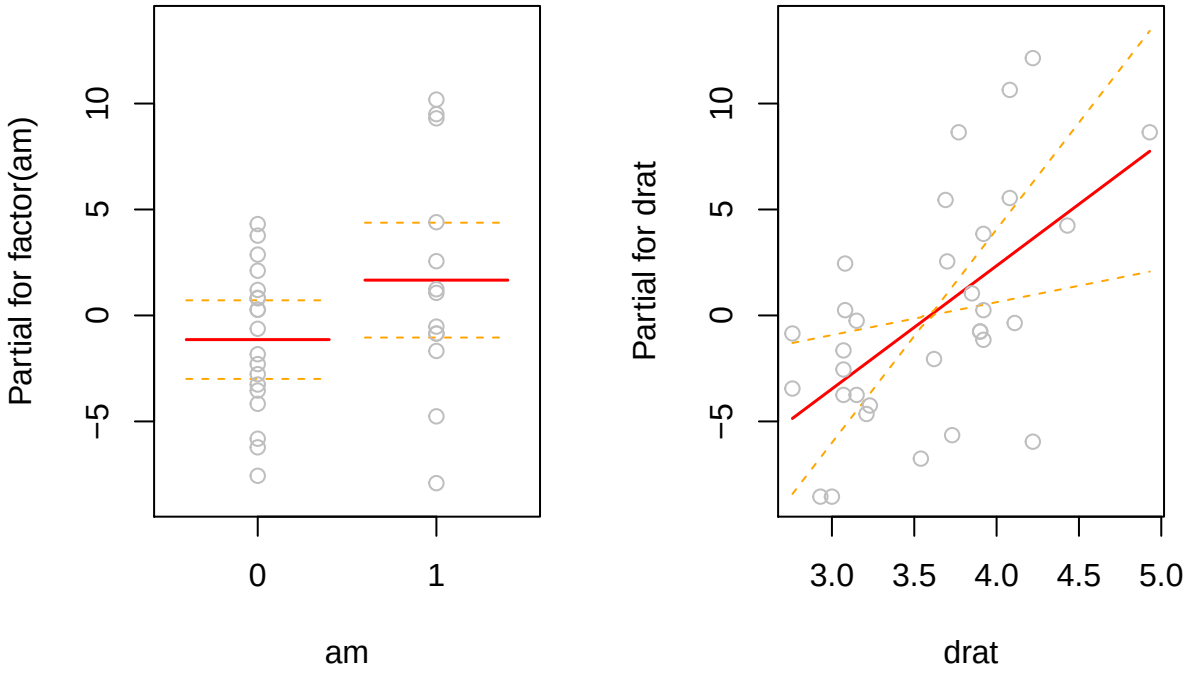
```
termplot(fit0,partial.resid = TRUE,se=TRUE)
```




```
par(mfrow=c(1,2))
termplot(fit1,partial.resid = TRUE,se=TRUE)
```




```
termplot(fit2,partial.resid = TRUE,se=TRUE)
```



Example: Boston Housing data

Youtube animation (*Boston-area housing price animated heat-map*)

La libraría `MASS` contiene el conjunto de datos de `Boston`, que registra el `medv` (valor mediano de la casa) de 506 vecindarios alrededor de Boston. Trataremos de predecir el `medv` usando 13 predictores tales como `rm` (número promedio de habitaciones por casa), `age` (edad promedio de las casas), y 'lstat (porcentaje de hogares con bajo estatus socioeconómico).

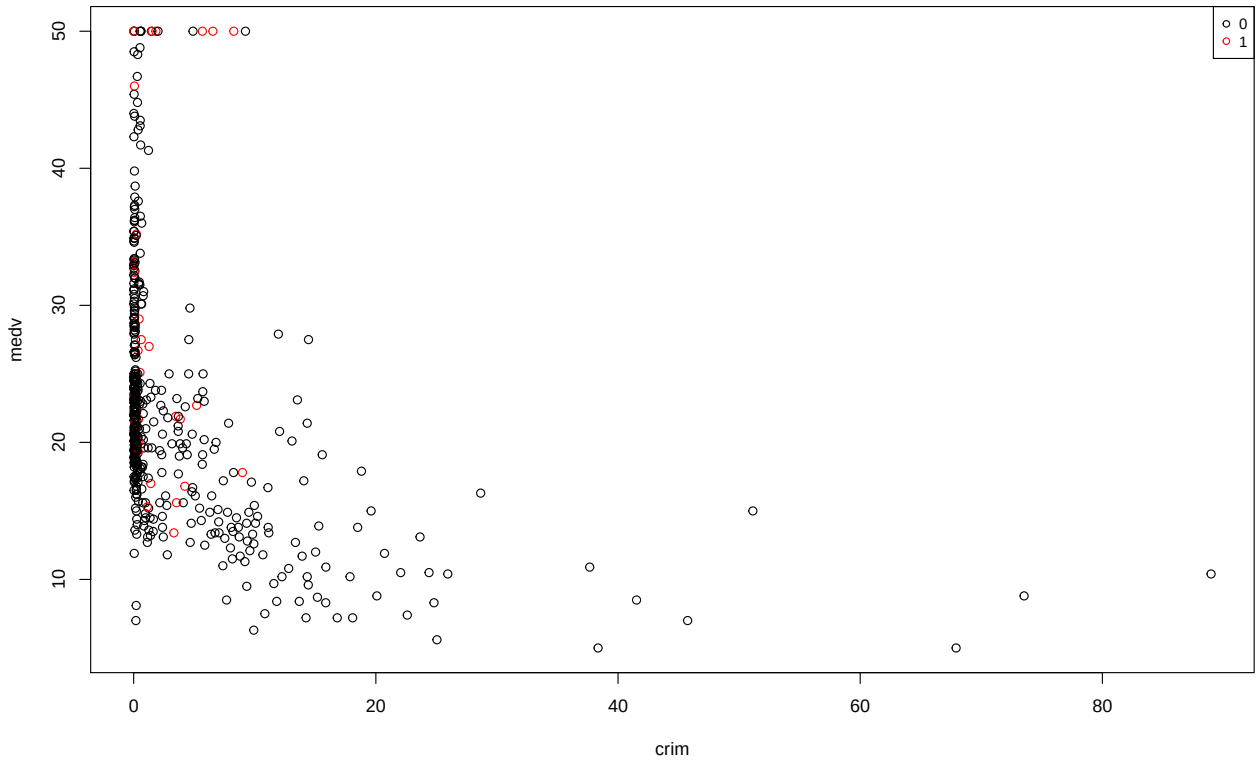
```r
library(MASS)
data("Boston")
?Boston
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

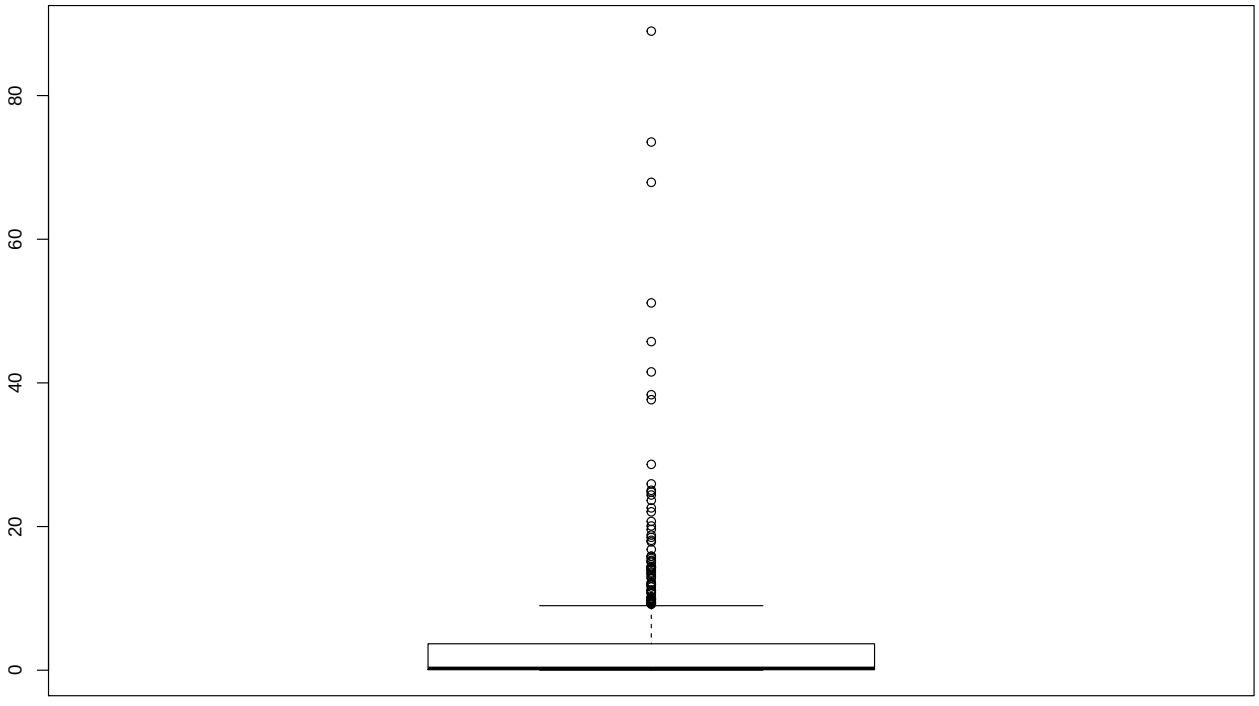
```r
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

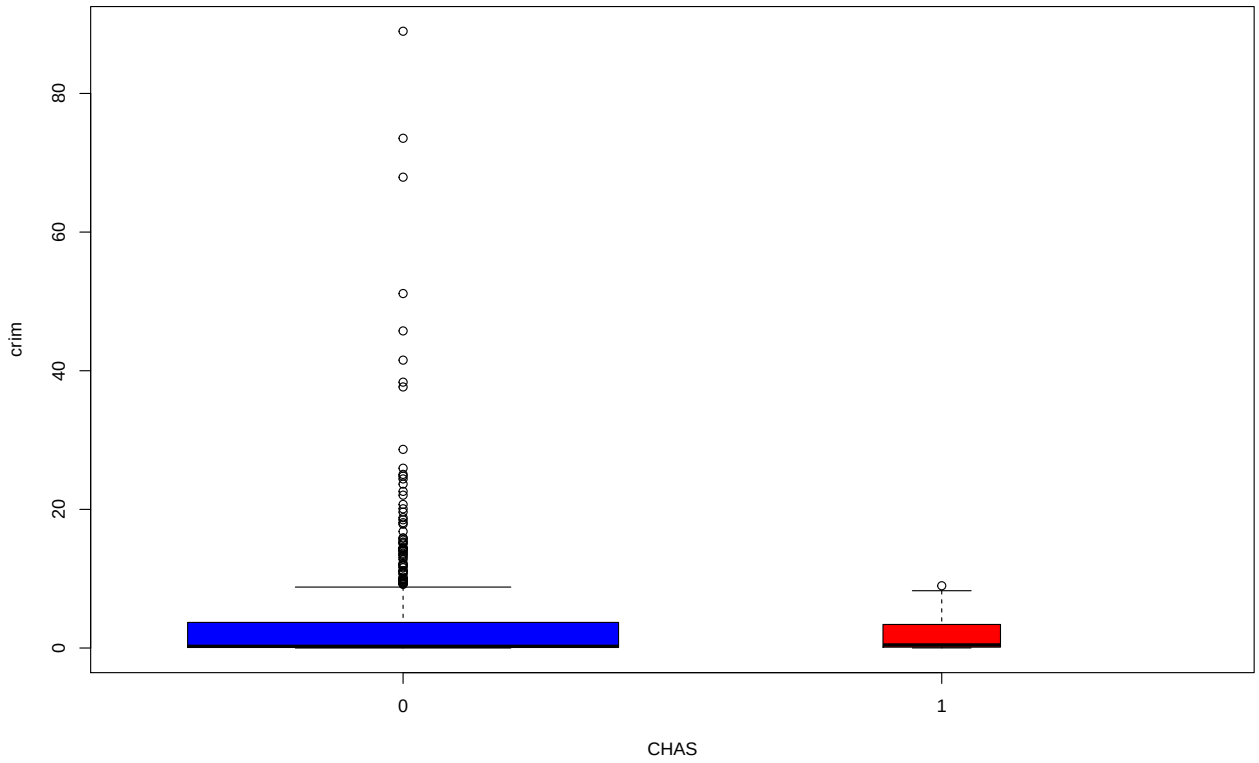
```r
# Some plots

attach(Boston)
plot(crim,medv,col=1+chas)
legend('topright', legend = levels(factor(chas)), col = 1:2, cex = 0.8, pch = 1)
```



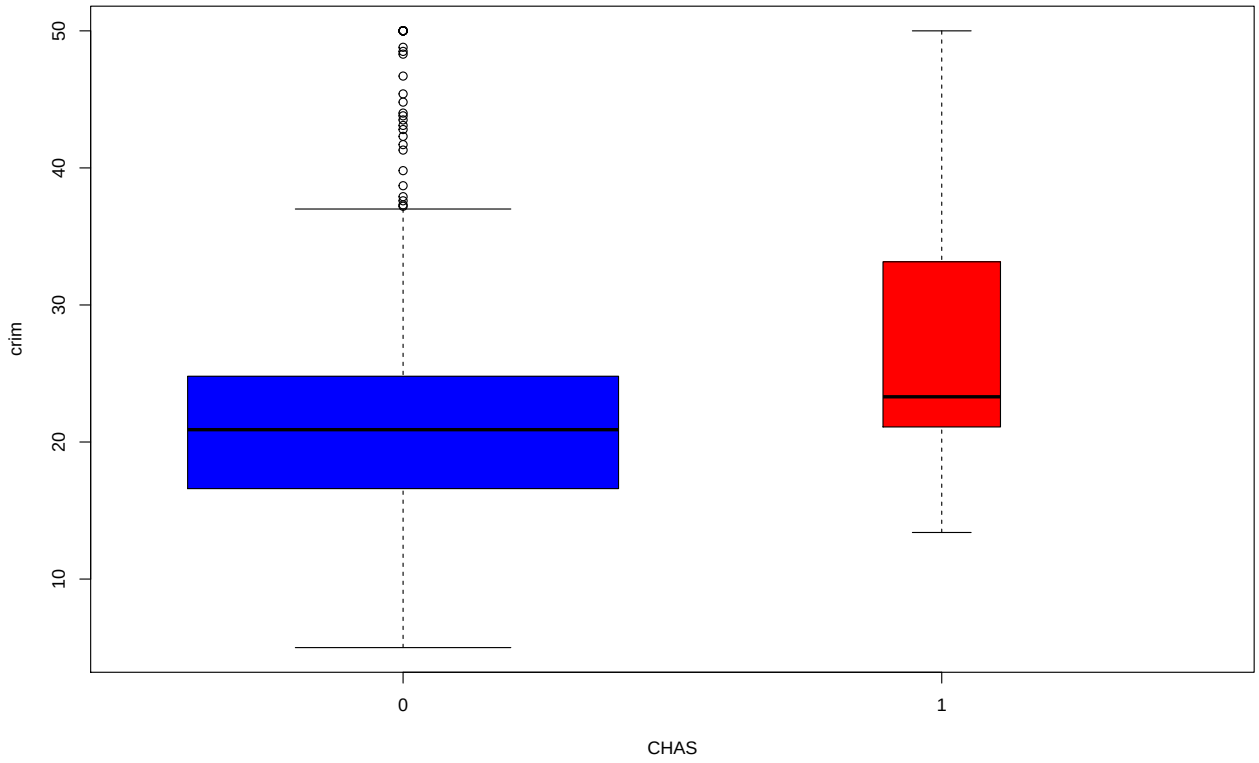

```
# boxplots
 boxplot(crim,data=Boston)
```



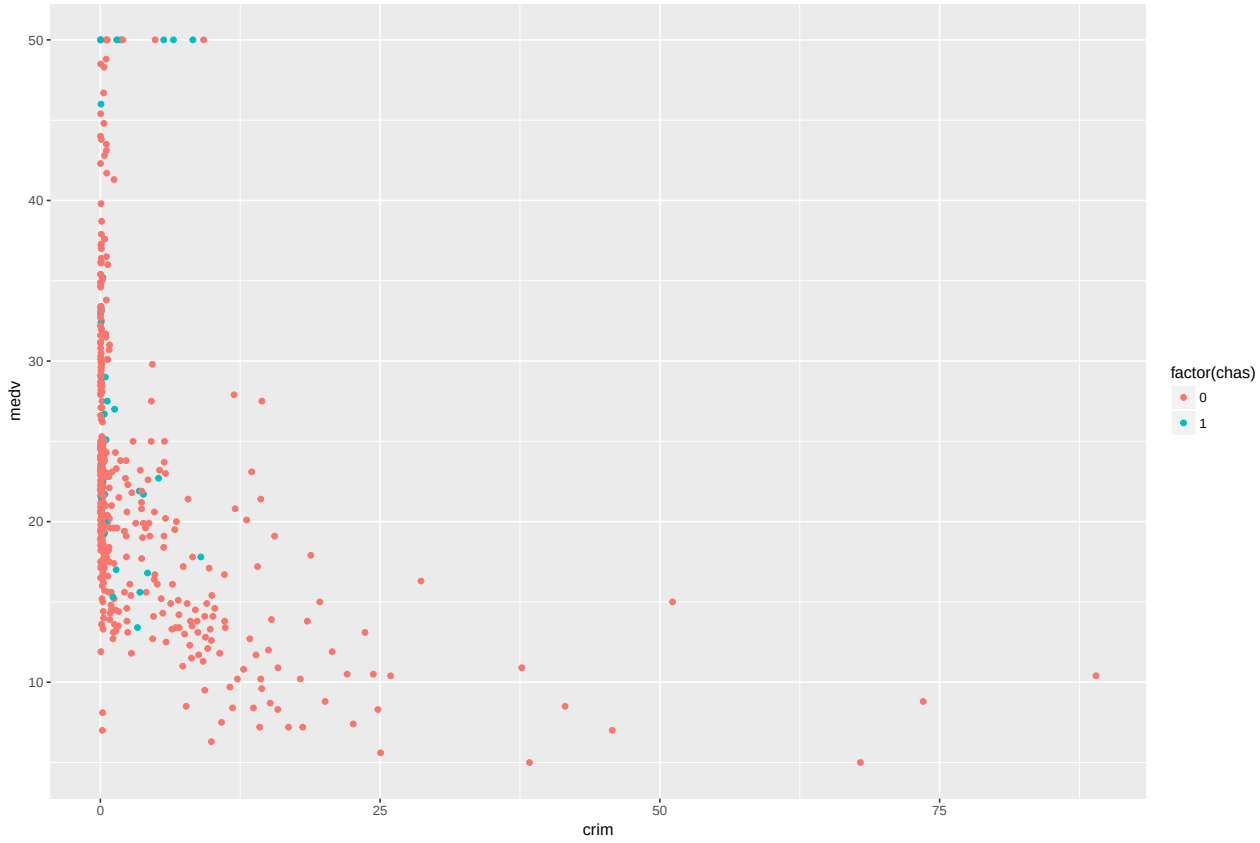

```
 boxplot(crim ~ factor(chas), data = Boston,xlab="CHAS",ylab="crim",col=c(4,2),varwidth=TRUE)
```



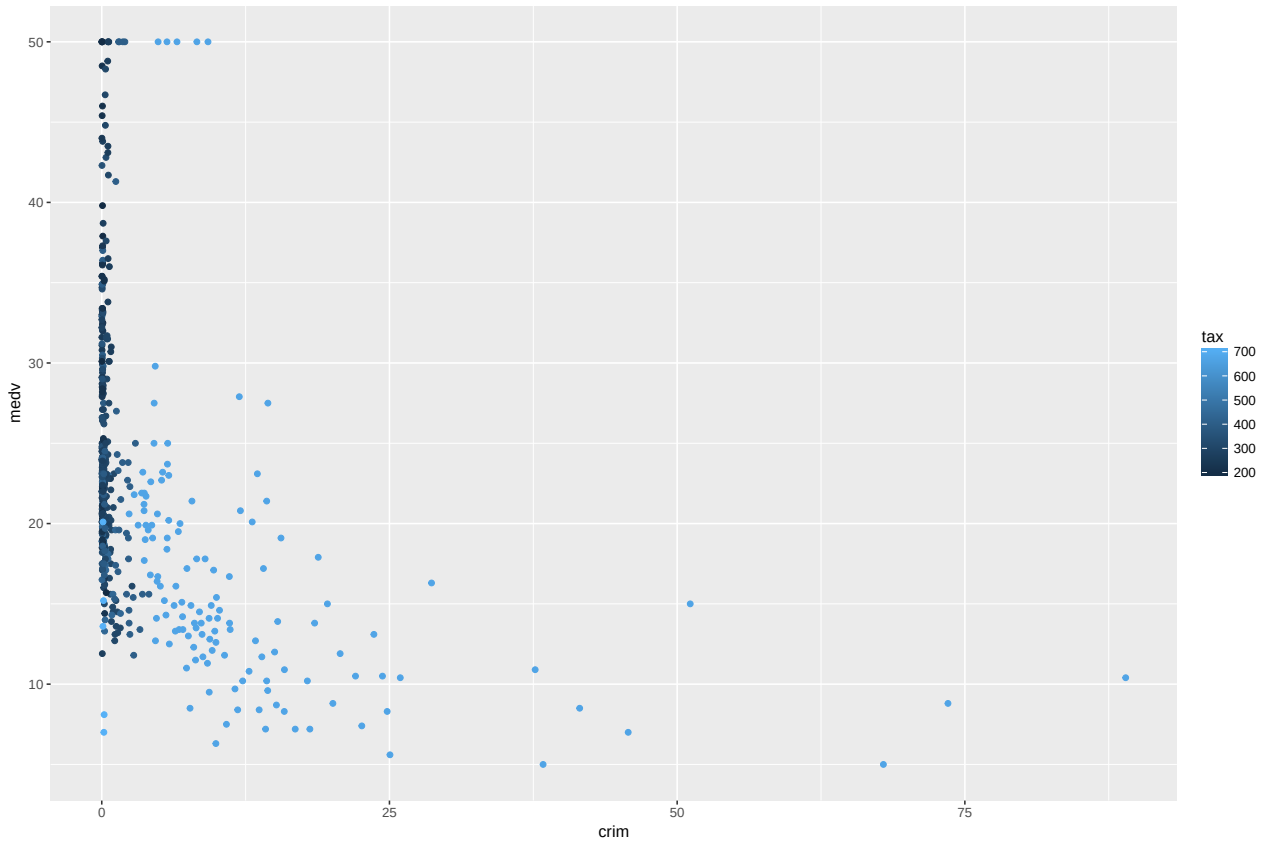

```
boxplot(medv ~ factor(chas), data = Boston,xlab="CHAS",ylab="crim",col=c(4,2),varwidth=TRUE)
```



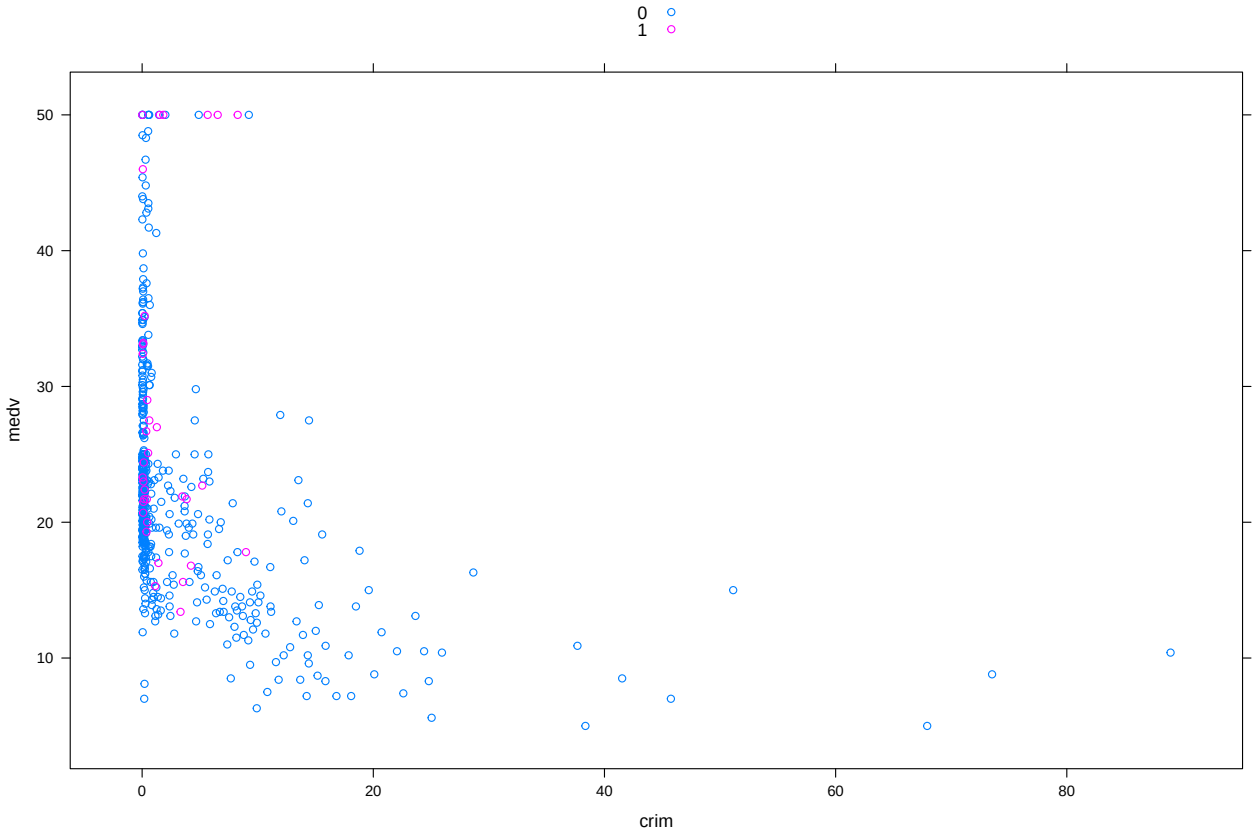

```
library(ggplot2)

qplot(crim,medv,data=Boston, colour=factor(chas))
```



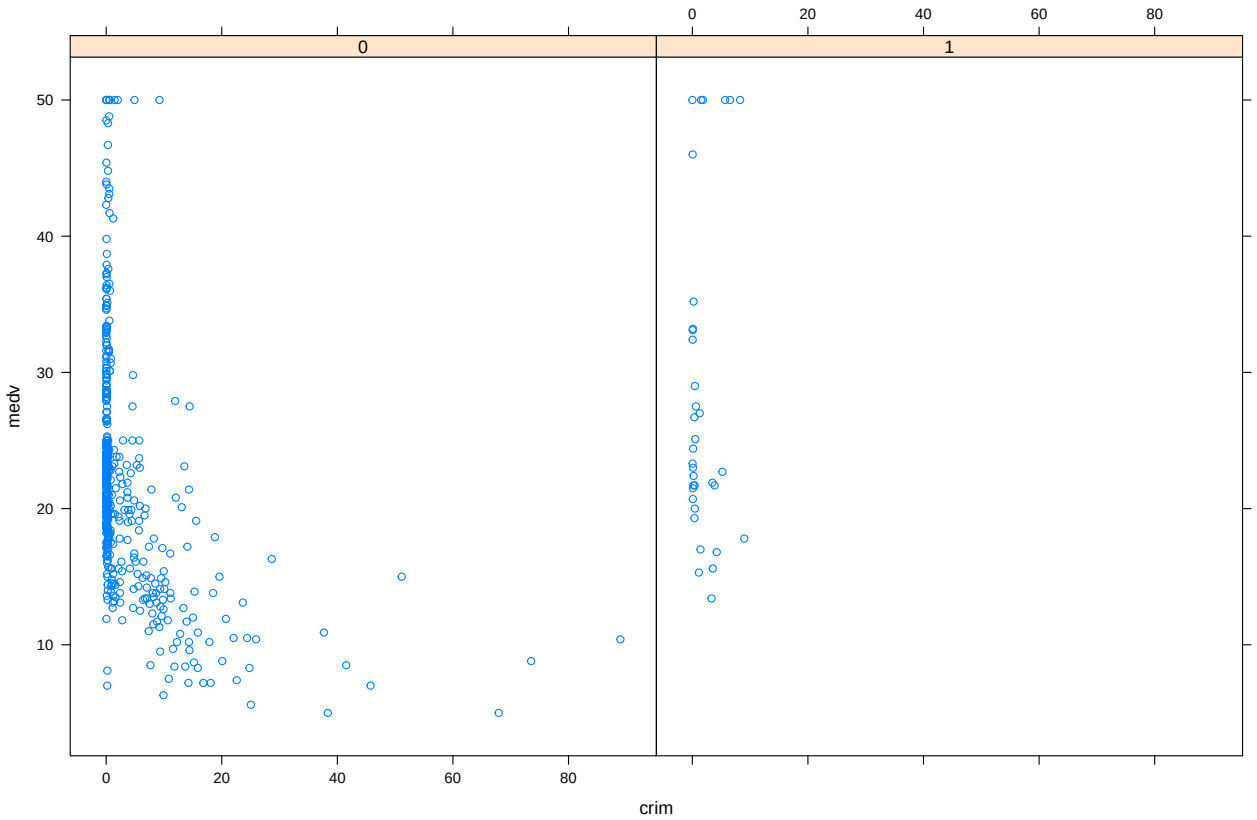

```
qplot(crim,medv,data=Boston, colour=tax)
```




```
library(lattice)
xyplot(medv~crim,groups=factor(chas),auto.key = TRUE,data=Boston)
```




```
xyplot(medv~crim|factor(chas),auto.key = TRUE,data=Boston)
```



Comenzaremos usando la función `lm()` para ajustar un modelo de regresión lineal simple, con `medv` como variable respuesta y `lstat` como predictor.

```
lm.fit <- lm(medv ~ lstat, data=Boston)
```

```
lm.fit
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

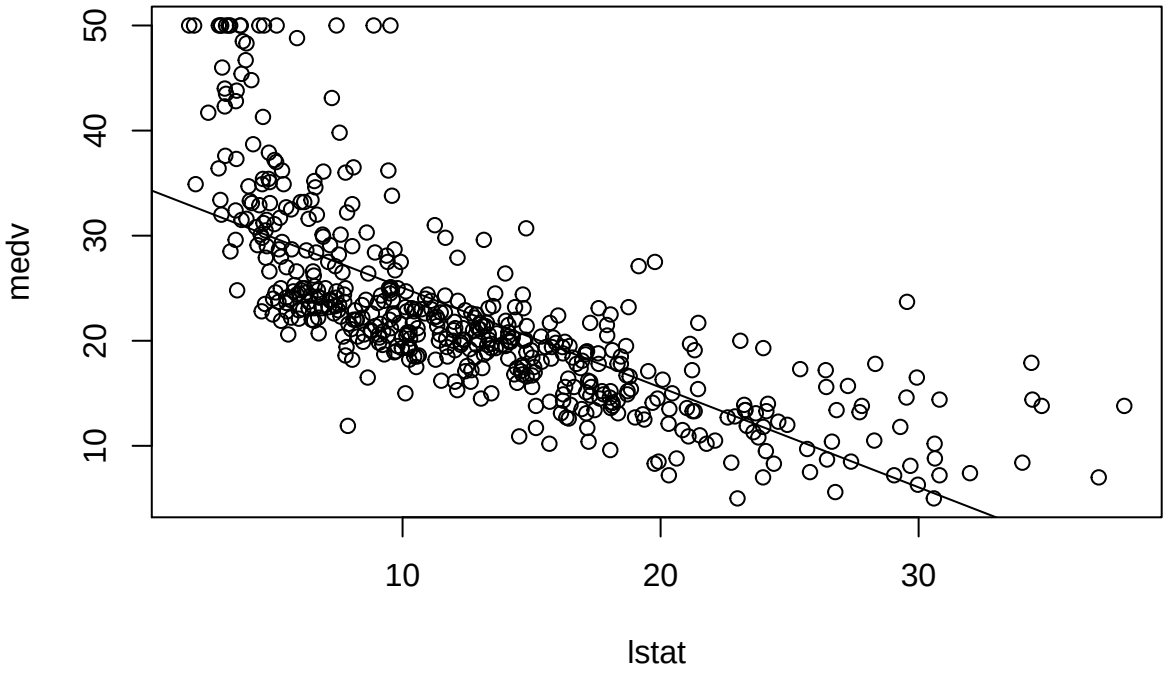
```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

Ahora dibujaremos `medv` y `lstat` junto con la línea de regresión de mínimos cuadrados usando las funciones `plot()` y `abline()`.

```
plot(medv ~ lstat, data = Boston)
abline(lm.fit)
```



Para obtener un intervalo de confianza para las estimaciones del coeficiente, podemos usar el comando `confint()`.

```
confint(lm.fit, level = 0.95)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

Consideramos la posibilidad de construir un intervalo de confianza para β_1 utilizando la información proporcionada por el resumen de `lm.fit`:

```
summary(lm.fit)$coefficients
```

```
##               Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) 34.5538409 0.56262735  61.41515 3.743081e-236
## lstat       -0.9500494 0.03873342 -24.52790  5.081103e-88
```

La función `predict()` puede ser usada para producir intervalos de confianza e intervalos de predicción para la predicción de `medv` para un valor dado de `lstat`.

```
CI <- predict(object = lm.fit, newdata = data.frame(lstat = c(5, 10, 15)),
              interval = "confidence")
CI
```

```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

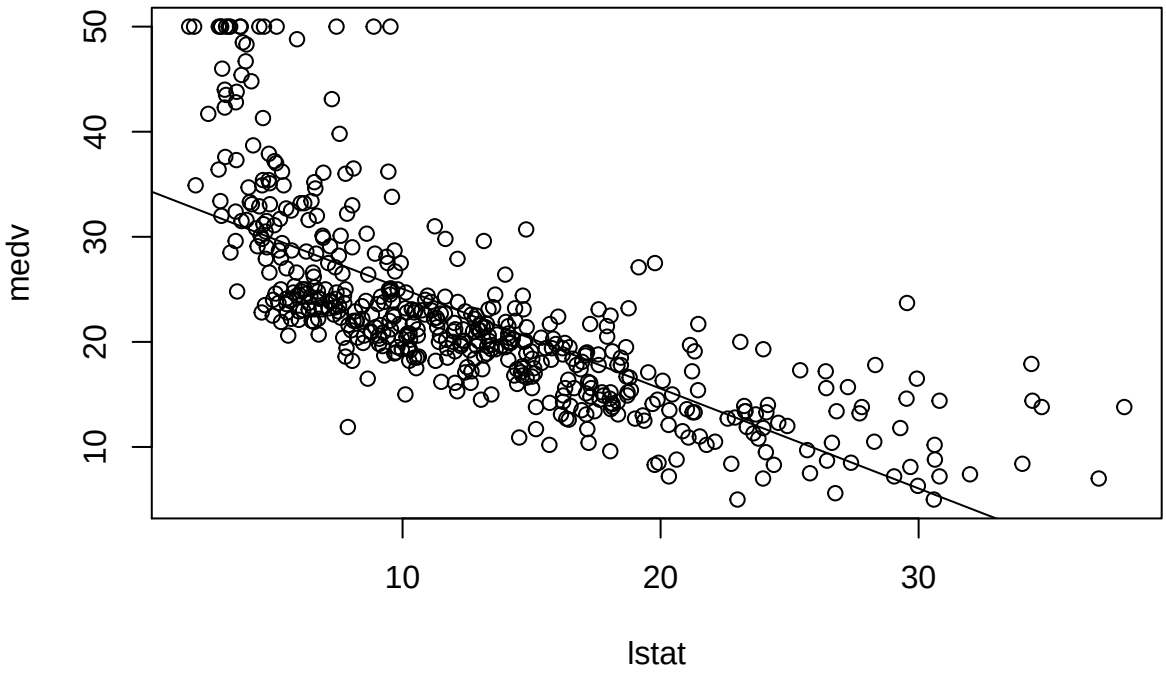
```
PI <- predict(object = lm.fit, newdata = data.frame(lstat = c(5, 10, 15)),
              interval = "predict")
PI
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

Por ejemplo, el intervalo de confianza del 95% asociado con un valor `lstat` de 10 es $(24.474132, 25.6325627)$ y el intervalo de predicción del 95% es $(12.8276263, 37.2790683)$. Como se esperaba, los intervalos de confianza y predicción se centran alrededor del mismo punto (un valor predicho de `25.0533473` para `medv cuando`lstat‘ es igual a 10), pero estos últimos son sustancialmente más amplios.

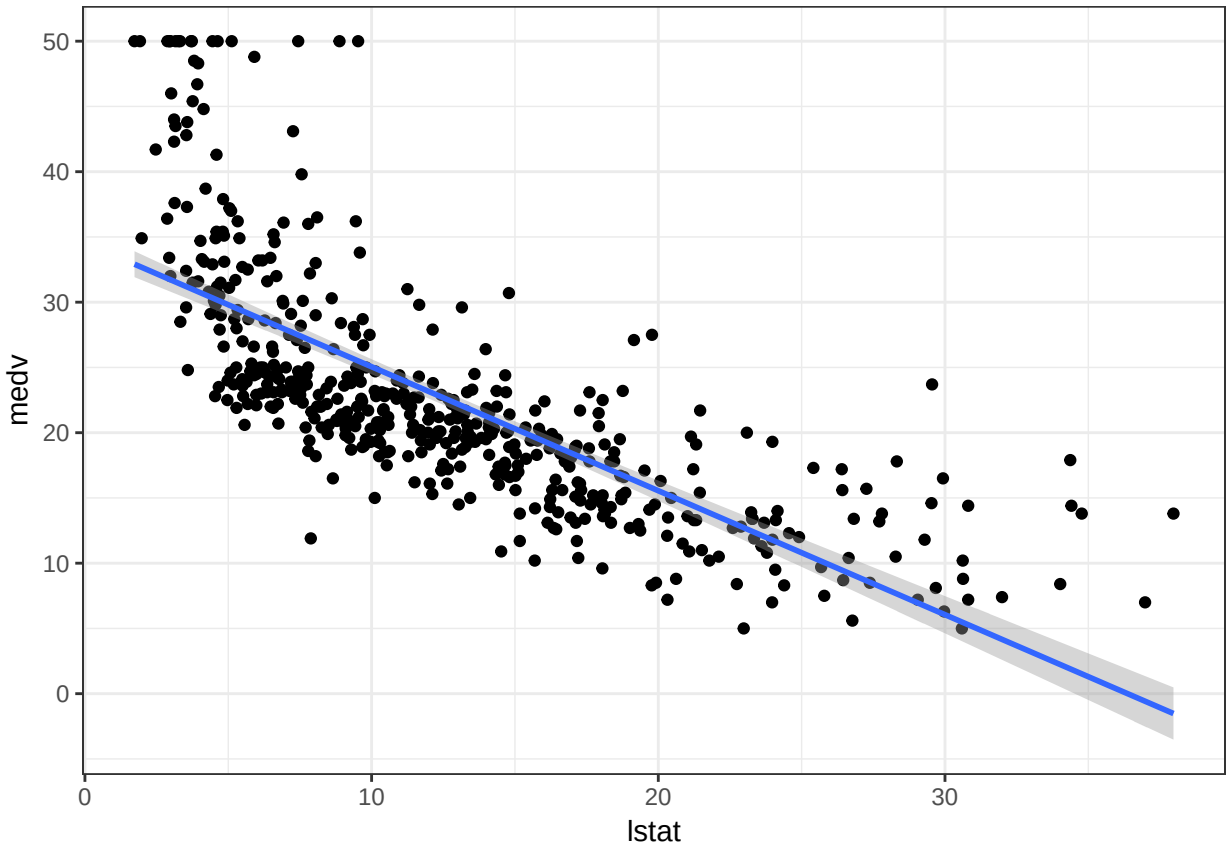
Ahora trazaremos `medv` y `lstat` junto con la línea de regresión de mínimos cuadrados usando las funciones `plot()` y `abline()`.

```
plot(medv ~ lstat, data = Boston)
abline(lm.fit)
```



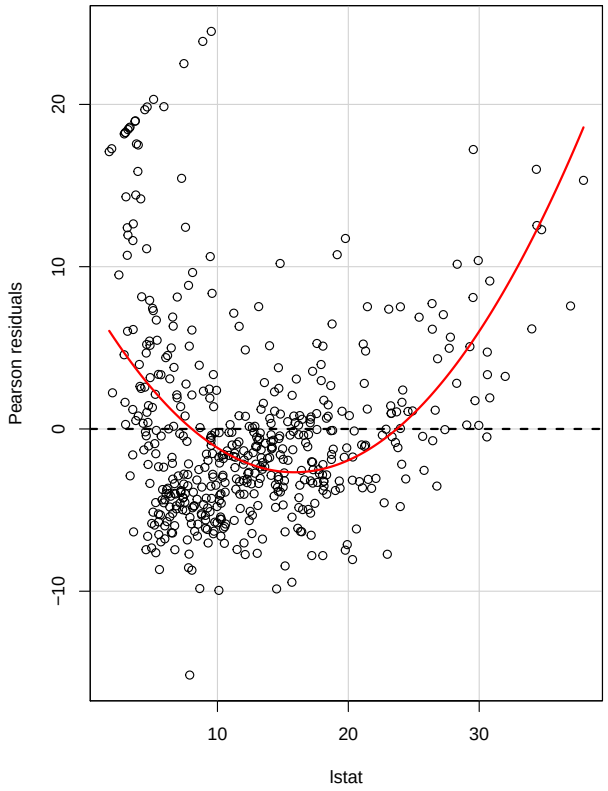
Con ggplot2

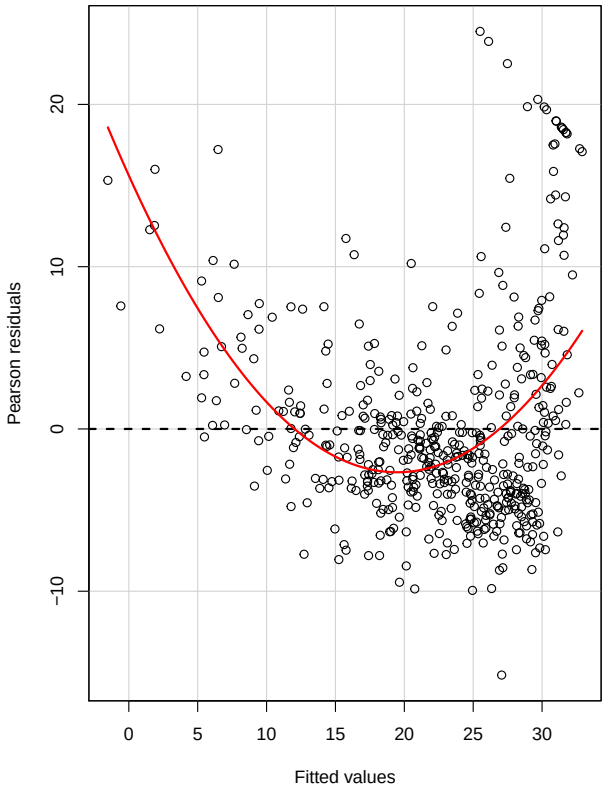
```
library(ggplot2)
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_bw()
```



La librería `car` tiene una función `residualPlots` para evaluar los residuos (calcula una prueba de curvatura para cada una de las parcelas añadiendo un término cuadrático y probando que la cuadrática sea cero). Ver `?residualPlots`

```
library(car)
residualPlots(lm.fit)
```






```
##            Test stat Pr(>|t|)
## lstat         11.628        0
## Tukey test    11.628        0
```

Regresión Lineal Múltiple

Para encajar un modelo de regresión lineal múltiple usando mínimos cuadrados, volvemos a usar la función `lm()`. La sintaxis `lm(y ~ x1 + x2 + x3)` se usa para encajar un modelo con tres predictores, `x1`, `x2`, y `x3`. La función `summary()` ahora produce los coeficientes de regresión para todos los predictores.

```
ls.fit <- lm(medv ~ lstat + age, data = Boston)
summary(ls.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

El conjunto de datos `Boston` contiene 13 variables, por lo que sería engorroso tener que escribirlas todas para poder realizar una regresión usando todos los predictores. En su lugar, podemos utilizar la siguiente abreviatura:

```
Boston$chas <- as.factor(Boston$chas)
ls.fit <- lm(medv ~ ., data = Boston)
summary(ls.fit)
```

```
##
## 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
## chas1        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
```

Podemos acceder a los componentes individuales de un objeto de resumen por nombre (escriba `?summary.lm` para ver lo que está disponible). Por lo tanto `summary(lm.fit)$r.sq` nos da los R^2, y `summary(lm.fit)$sigma` nos da $\hat{sigma}$.

Si queremos realizar una regresión usando todas las variables pero excepto una, podemos eliminarla usando `-`. Por ejemplo, en la salida de regresión anterior, `age` tiene un alto p-valor. Así que tal vez queramos hacer una regresión excluyendo este predictor. La siguiente sintaxis resulta en una regresión usando todos los predictores excepto `age`.

```
ls.fit1 <- lm(medv ~ . - age, data = Boston)
summary(ls.fit1)
```

```
##
## Call:
```

```
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas1         2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

Términos de interacción

Es fácil incluir términos de interacción en un modelo lineal usando la función `lm()`. La sintaxis `lstat:black` le dice a `R` que incluya un término de interacción entre `lstat` y `black`. La sintaxis `lstat*age` incluye simultáneamente `lstat`,`age`, y el término de interacción `lstat` ×`age` como predictores; es una abreviatura de `lstat + age + lstat:age`.

```
summary(lm(medv ~ lstat*age, data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
```

```
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

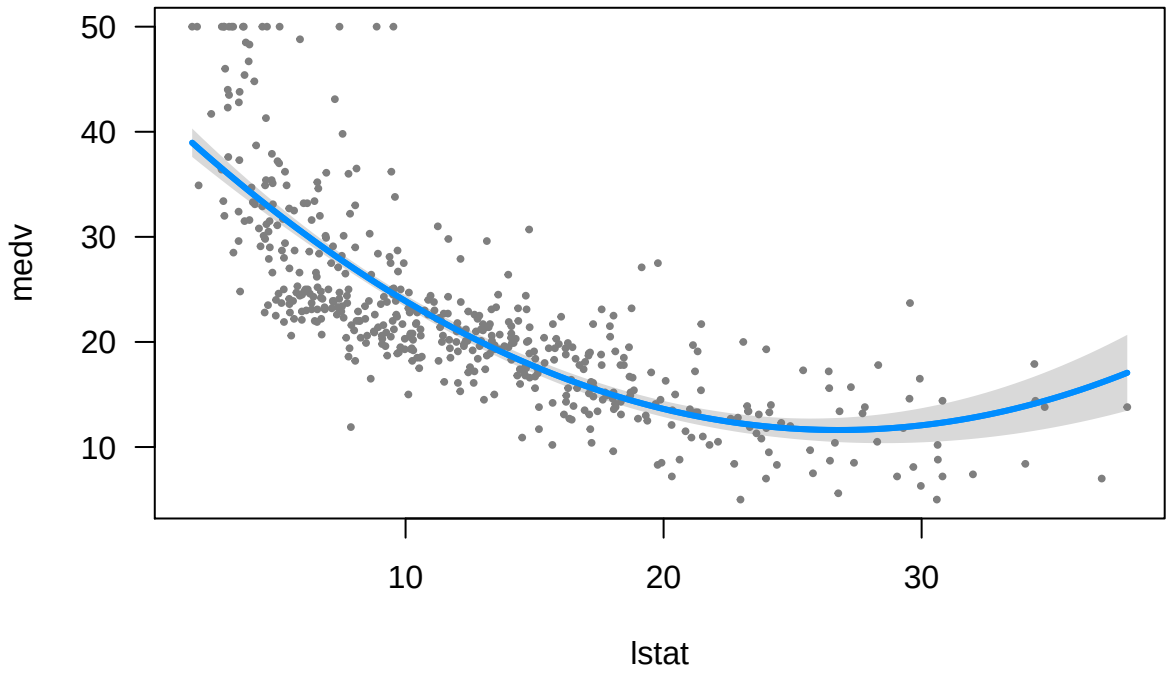
Transformaciones no lineales para las variables predictoras

La función `lm()` también puede acomodar transformaciones no lineales de los predictores. Por ejemplo, dado un predictor X podemos crear un predictor X^2 usando `I(X^2)`. La función `I()` es necesaria ya que `^` tiene un significado especial en una fórmula; envolviendo como lo hacemos permite el uso estándar en `R`, que es `I()` para elevar X a la potencia 2. Ahora realizamos una regresión de `medv` sobre `lstat` y `lstat`^{2} .

```
lm.fit2 <- lm(medv ~ lstat + I(lstat^2), data = Boston)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
# plot
library(visreg)
visreg(lm.fit2)
```



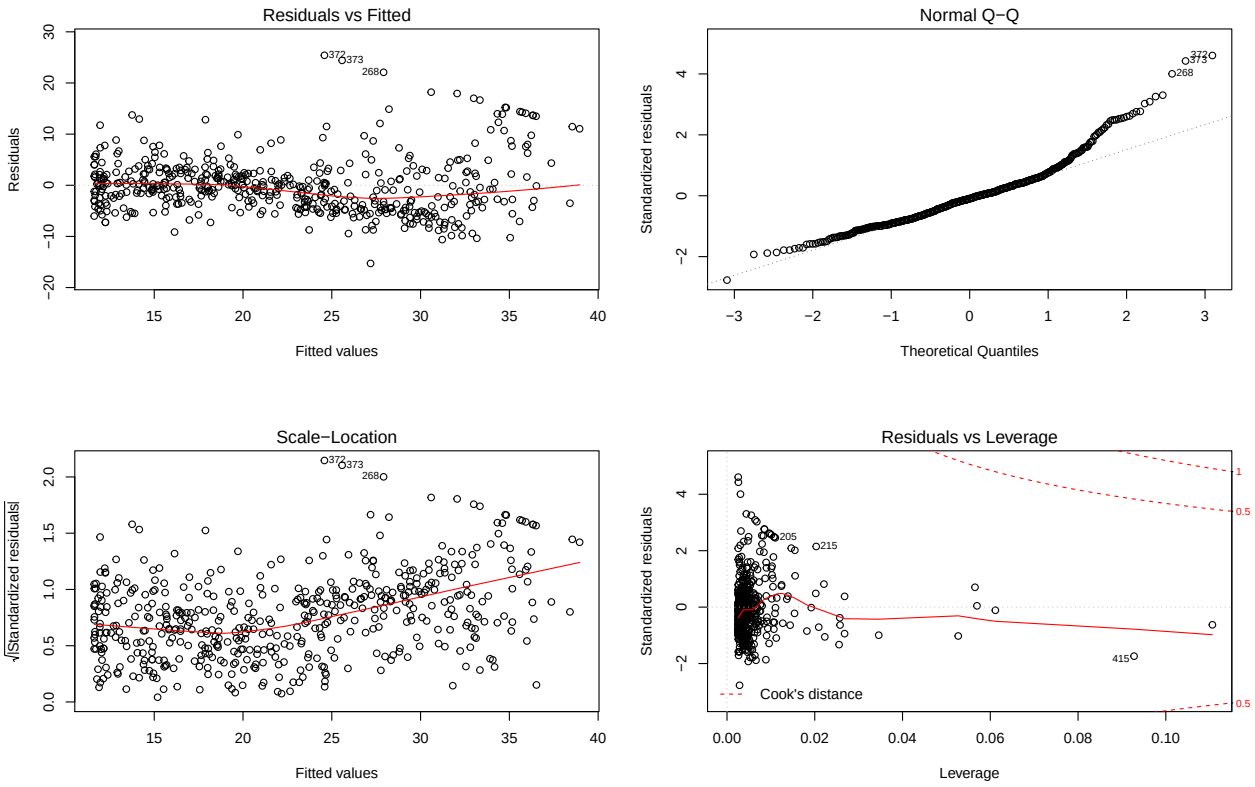
El p-valor cercano a cero asociado con el término cuadrático sugiere que conduce a un modelo mejorado. Usamos la función `anova()` para cuantificar aún más hasta qué punto el ajuste cuadrático es superior al ajuste lineal.

```
anova(lm.fit, lm.fit2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + I(lstat^2)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
## 1    504 19472
## 2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Aquí el Modelo 1 (`lm.fit`) representa el submodelo lineal que contiene sólo un predictor, `lstat`, mientras que el Modelo 2 (`lm.fit2`) corresponde al modelo cuadrático más grande que tiene dos predictores, `lstat` y `I(lstat^2)`. La función `anova()` realiza una prueba de hipótesis comparando los dos modelos. La hipótesis nula es que los dos modelos se ajustan a los datos igualmente bien, y la hipótesis alternativa es que el modelo completo es superior. Aquí la estadística F es `135.1998221` y el valor p asociado es virtualmente cero. Esto proporciona una evidencia muy clara de que el modelo que contiene los predictores `lstat` y `I(lstat^2)` es muy superior al modelo que sólo contiene el predictor `lstat`. Esto no es sorprendente, ya que antes vimos evidencia de no linealidad en la relación entre `medv` y `lstat`. Si escribimos

```
par(mfrow = c(2,2))
plot(lm.fit2)
```




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

entonces vemos que cuando el término `I(lstat^2)` se incluye en el modelo, hay poco patrón discernible en los residuos.

Para crear un ajuste cúbico, podemos incluir un predictor de la forma `I(X^3)`. Sin embargo, este enfoque puede empezar a ser engorroso para los polinomios de orden superior. Un mejor enfoque implica usar la función `poly()` para crear el polinomio dentro de `lm()`. Por ejemplo, el siguiente comando produce un ajuste polinómico de quinto orden:

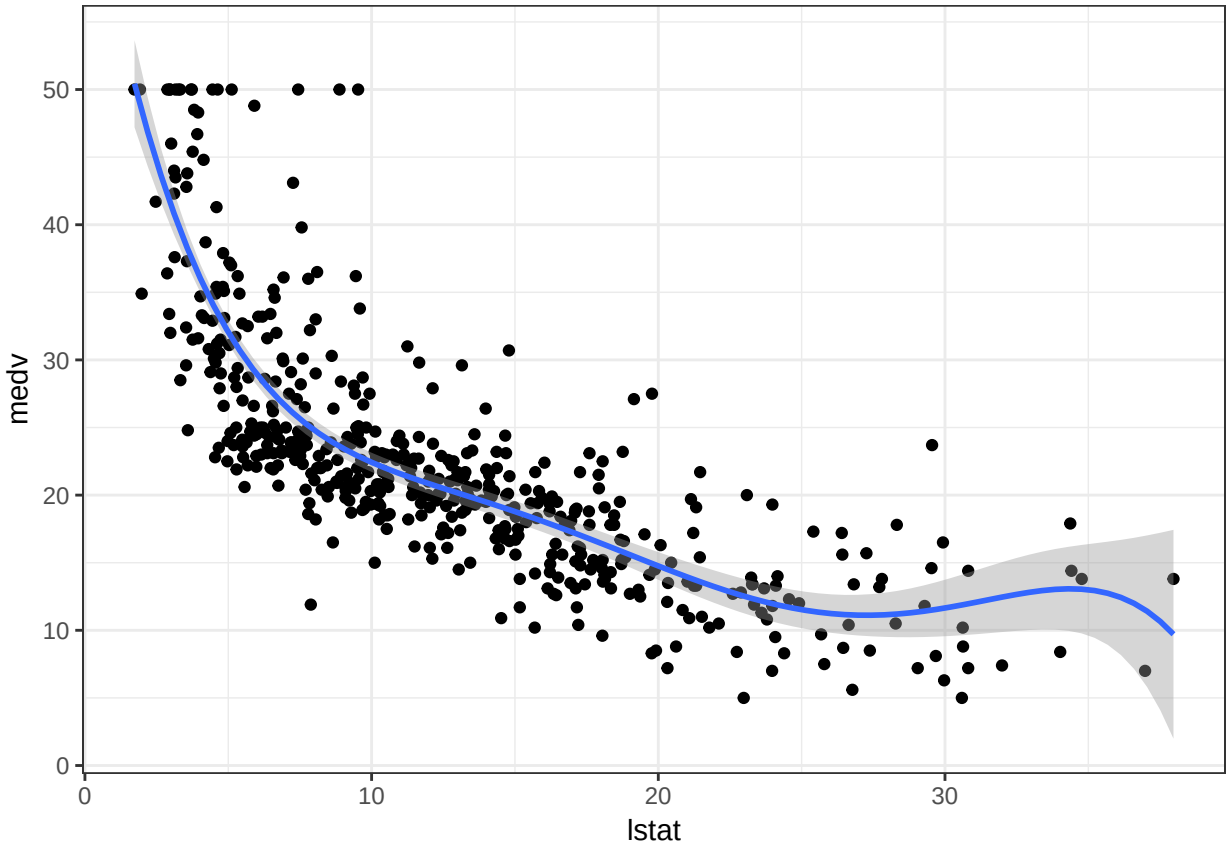
```
lm.fit5 <- lm(medv ~ poly(lstat, 5), data = Boston)
summary(lm.fit5)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5), data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

Esto sugiere que incluir términos polinómicos adicionales, hasta el quinto orden, conduce a una mejora en el ajuste del modelo. Sin embargo, la investigación adicional de los datos revela que ningún término polinómico más allá del quinto orden tiene p-valores significativos en un ajuste de regresión.

```
library(ggplot2)
ggplot(data = Boston, aes(x = lstat, y = medv)) +
  geom_point() +
  theme_bw() +
  stat_smooth(method = "lm", formula = y ~ poly(x, 5))
```



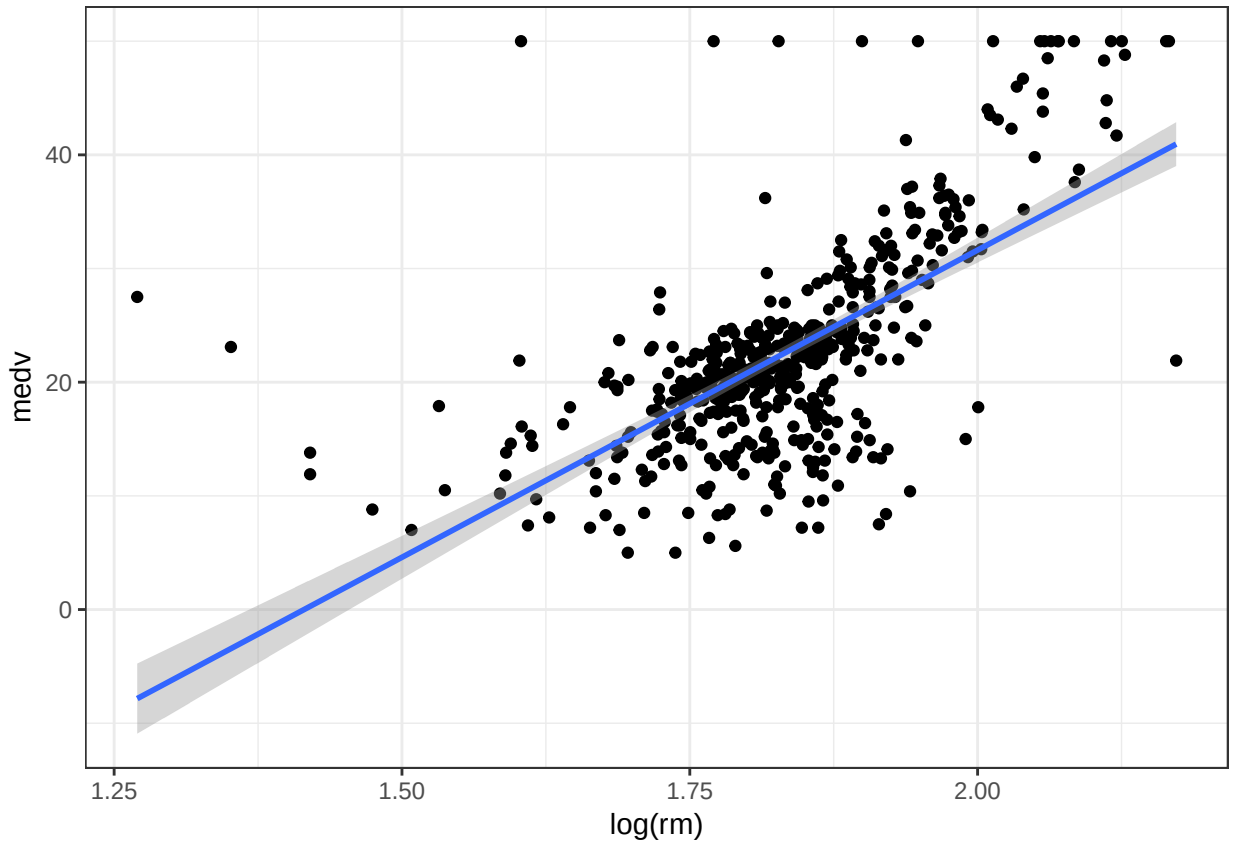
Por supuesto, no estamos de ninguna manera restringidos a usar transformaciones polinómicas de los predictores. Aquí probamos una transformación logarítmica de la variable respuesta.

```
summary(lm(medv ~ log(rm), data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ log(rm), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.487  -2.875  -0.104   2.837  39.816
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -76.488      5.028  -15.21   <2e-16 ***
## log(rm)       54.055      2.739   19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.915 on 504 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4347
## F-statistic: 389.3 on 1 and 504 DF,  p-value: < 2.2e-16
```

```
ggplot(data = Boston, aes(x = log(rm), y = medv)) +
  geom_point() +
  theme_bw() +
  stat_smooth(method = "lm")
```



Selección de modelos: librería `leaps`

```
library(leaps)
leaps <- regsubsets(medv~.,data=Boston,nbest=10)
summary(leaps)
```

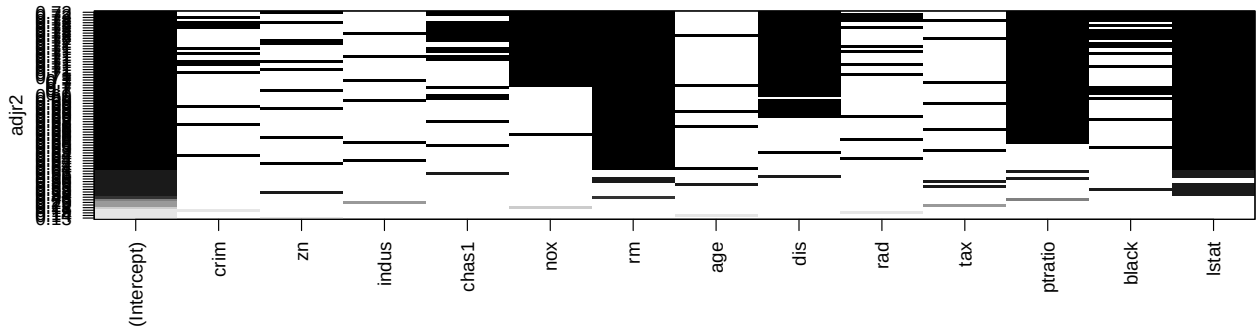
```
## Subset selection object
## Call: regsubsets.formula(medv ~ ., data = Boston, nbest = 10)
## 13 Variables  (and intercept)
##         Forced in Forced out
## crim        FALSE      FALSE
```

```
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas1       FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## black       FALSE      FALSE
## lstat       FALSE      FALSE
## 10 subsets of each size up to 8
## Selection Algorithm: exhaustive
##           crim zn  indus chas1 nox rm  age dis rad tax ptratio black lstat
## 1  ( 1 )  " "  " " " " " "   " "   " " " " " " " " " " " " " "     " "   "*"
## 1  ( 2 )  " "  " " " " " "   " "   " " "*" " " " " " " " " " " " " "     " "   " "
## 1  ( 3 )  " "  " " " " " "   " "   " " " " " " " " " " " " " "*"     " "   " "
## 1  ( 4 )  " "  " " "*" " "   " "   " " " " " " " " " " " " " " "     " "   " "
## 1  ( 5 )  " "  " " " " " "   " "   " " " " " " " " " " "*" " " "     " "   " "
## 1  ( 6 )  " "  " " " " " "   " "   "*" " " " " " " " " " " " " "     " "   " "
## 1  ( 7 )  "*"  " " " " " "   " "   " " " " " " " " " " " " " " "     " "   " "
## 1  ( 8 )  " "  " " " " " "   " "   " " " " " " " " "*" " " " " "     " "   " "
## 1  ( 9 )  " "  " " " " " "   " "   " " " " "*" " " " " " " " " "     " "   " "
## 1  ( 10 ) " "  "*" " " "   " "   " " " " " " " " " " " " " " " "     " "   " "
## 2  ( 1 )  " "  " " " " " "   " "   " " "*" " " " " " " " " " " "     " "   "*"
## 2  ( 2 )  " "  " " " " " "   " "   " " " " " " " " " " " " "*"     " "   "*"
## 2  ( 3 )  " "  " " " " " "   "*"   " " " " " " " " " " " " " "     " "   "*"
## 2  ( 4 )  " "  " " " " " "   " "   " " " " " " "*" " " " " " "     " "   "*"
## 2  ( 5 )  " "  " " " " " "   " "   " " "*" " " " " " " " " "*"     " "   " "
## 2  ( 6 )  " "  " " " " " "   " "   " " "*" " " " " " " " "*" " "     " "   " "
## 2  ( 7 )  " "  " " " " " "   " "   " " " " "*" " " " " " " " "     " "   "*"
## 2  ( 8 )  " "  " " " " " "   " "   " " " " " " " " " " "*" " "     " "   "*"
## 2  ( 9 )  " "  " " " " " "   " "   " " " " " " " " " " " " " "     "*"   "*"
## 2  ( 10 ) " "  "*" " " "   " "   " " " " " " " " " " " " " "     " "   "*"
## 3  ( 1 )  " "  " " " " " "   " "   " " "*" " " " " " " " " "*"     " "   "*"
## 3  ( 2 )  " "  " " " " " "   "*"   " " "*" " " " " " " " " " "     " "   "*"
## 3  ( 3 )  " "  " " " " " "   " "   " " "*" " " " " " " " " " "     "*"   "*"
## 3  ( 4 )  " "  " " " " " "   " "   " " "*" " " " " " " "*" " "     " "   "*"
## 3  ( 5 )  " "  " " " " " "   " "   " " "*" " " "*" " " " " " "     " "   "*"
## 3  ( 6 )  "*"  " " " " " "   " "   " " "*" " " " " " " " " " "     " "   "*"
## 3  ( 7 )  " "  " " " " " "   " "   " " "*" " " " " "*" " " " "     " "   "*"
## 3  ( 8 )  " "  " " "*" " "   " "   " " "*" " " " " " " " " " "     " "   "*"
## 3  ( 9 )  " "  "*" " " "   " "   " " "*" " " " " " " " " " " "     " "   "*"
## 3  ( 10 ) " "  " " " " " "   " "   " " "*" "*" " " " " " " " " "     " "   "*"
## 4  ( 1 )  " "  " " " " " "   " "   " " "*" " " "*" " " " " "*"     " "   "*"
## 4  ( 2 )  " "  " " " " " "   " "   " " "*" " " " " " " " " "*"     "*"   "*"
## 4  ( 3 )  " "  " " " " " "   "*"   " " "*" " " " " " " " " "*"     " "   "*"
## 4  ( 4 )  "*"  " " " " " "   " "   " " "*" " " " " " " " " "*"     " "   "*"
## 4  ( 5 )  " "  " " " " " "   " "   " " "*" "*" " " " " " " "*"     " "   "*"
## 4  ( 6 )  " "  " " " " " "   " "   " " "*" " " " " " " "*" "*"     " "   "*"
## 4  ( 7 )  " "  " " " " " "   " "   "*" "*" " " " " " " " "*"     " "   "*"
## 4  ( 8 )  " "  "*" " " "   " "   " " "*" " " " " " " " " "*"     " "   "*"
## 4  ( 9 )  " "  " " " " " "   " "   " " "*" " " " " " "*" " " "*"     " "   "*"
```

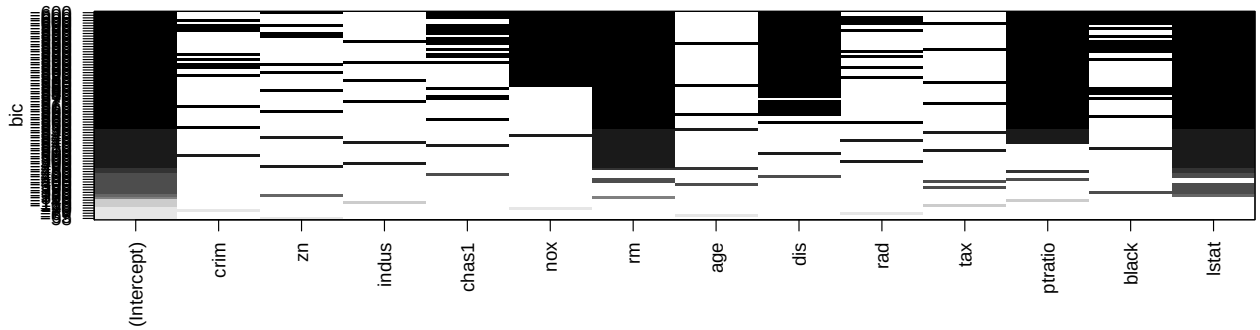
```
## 4  ( 10 ) " "  " " "*"   " "   " " "*" " " " " " " " " " "*"     " "   "*"
## 5  ( 1 )  " "  " " " "   " "   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 5  ( 2 )  " "  " " " "   " "   " " "*" " " "*" " " " " " "*"     "*"   "*"
## 5  ( 3 )  " "  " " " "   "*"   " " "*" " " "*" " " " " " "*"     " "   "*"
## 5  ( 4 )  " "  " " " "   "*"   " " "*" " " " " " " " " "*"     "*"   "*"
## 5  ( 5 )  " "  " " "*"   " "   " " "*" " " "*" " " " " " "*"     " "   "*"
## 5  ( 6 )  " "  " " " "   " "   " " "*" " " "*" " " "*" "*"     " "   "*"
## 5  ( 7 )  "*"  " " " "   " "   " " "*" " " "*" " " " " " "*"     " "   "*"
## 5  ( 8 )  " "  "*" " "   " "   " " "*" " " "*" " " " " " "*"     " "   "*"
## 5  ( 9 )  " "  " " " "   " "   " " "*" "*" "*" " " " " " "*"     " "   "*"
## 5  ( 10 ) " "  " " " "   " "   " " "*" " " "*" "*" " " " "*"     " "   "*"
## 6  ( 1 )  " "  " " " "   "*"   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 6  ( 2 )  " "  " " " "   " "   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 6  ( 3 )  " "  "*" " "   " "   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 6  ( 4 )  "*"  " " " "   " "   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 6  ( 5 )  " "  " " " "   " "   "*" "*" " " "*" "*" " " "*"     " "   "*"
## 6  ( 6 )  " "  " " "*"   " "   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 6  ( 7 )  " "  " " " "   " "   "*" "*" " " "*" " " "*" "*"     " "   "*"
## 6  ( 8 )  " "  " " " "   " "   "*" "*" "*" "*" " " " " " "*"     " "   "*"
## 6  ( 9 )  " "  " " " "   "*"   " " "*" " " "*" " " " " " "*"     "*"   "*"
## 6  ( 10 ) " "  "*" " "   " "   " " "*" " " "*" " " " " " "*"     "*"   "*"
## 7  ( 1 )  " "  " " " "   "*"   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 7  ( 2 )  " "  "*" " "   "*"   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 7  ( 3 )  " "  "*" " "   " "   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 7  ( 4 )  " "  " " " "   " "   "*" "*" " " "*" "*" " " " "*"     "*"   "*"
## 7  ( 5 )  "*"  " " " "   "*"   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 7  ( 6 )  " "  " " " "   "*"   "*" "*" " " "*" "*" " " " "*"     " "   "*"
## 7  ( 7 )  "*"  " " " "   " "   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 7  ( 8 )  " "  " " "*"   "*"   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 7  ( 9 )  "*"  "*" " "   " "   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 7  ( 10 ) "*"  " " " "   " "   "*" "*" " " "*" "*" " " " "*"     " "   "*"
## 8  ( 1 )  " "  "*" " "   "*"   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 8  ( 2 )  " "  " " " "   "*"   "*" "*" " " "*" "*" " " " "*"     "*"   "*"
## 8  ( 3 )  "*"  " " " "   " "   "*" "*" " " "*" "*" " " " "*"     "*"   "*"
## 8  ( 4 )  " "  " " " "   " "   "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 8  ( 5 )  "*"  "*" " "   "*"   "*" "*" " " "*" " " " " " "*"     " "   "*"
## 8  ( 6 )  "*"  " " " "   "*"   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 8  ( 7 )  "*"  " " " "   "*"   "*" "*" " " "*" "*" " " " "*"     " "   "*"
## 8  ( 8 )  " "  " " "*"   "*"   "*" "*" " " "*" " " " " " "*"     "*"   "*"
## 8  ( 9 )  " "  " " " "   "*"   "*" "*" "*" "*" " " " " " "*"     "*"   "*"
## 8  ( 10 ) " "  " " " "   "*"   "*" "*" " " "*" " " "*" "*"     "*"   "*"
```

Para ver los modelos clasificados de acuerdo con los criterios ajustados R-cuadrado y BIC, respectivamente:

```
plot(leaps, scale="adjr2")
```




```
plot(leaps, scale="bic")
```



Métodos automáticos son útiles cuando el número de variables explicativas es grande y no es posible ajustar todos los modelos posibles. En este caso, es más eficaz utilizar un algoritmo de búsqueda (p. ej., forward selection, backward elimination y stepwise regression) para encontrar el mejor modelo.

La función R `step()` sirve para realizar la selección de variables. Para realizar selección hacia adelante (forward selection) necesitamos empezar por especificando un modelo inicial y la gama de modelos que queremos examinar en la búsqueda.

```
null <- lm(medv ~ 1, data=Boston)
full <- lm(medv ~ ., data=Boston)
```

Esto le dice a R que comience con el modelo nulo y busque a través de los modelos que se encuentran en el rango entre el modelo nulo y el modelo completo usando el algoritmo de selección hacia adelante.

```
step(null, scope=list(lower=null, upper=full), direction="forward")
```

```
## Start:  AIC=2246.51
## medv ~ 1
##
##           Df Sum of Sq   RSS    AIC
## + lstat    1   23243.9 19472 1851.0
## + rm       1   20654.4 22062 1914.2
## + ptratio  1   11014.3 31702 2097.6
## + indus    1    9995.2 32721 2113.6
## + tax      1    9377.3 33339 2123.1
## + nox      1    7800.1 34916 2146.5
## + crim     1    6440.8 36276 2165.8
## + rad      1    6221.1 36495 2168.9
## + age      1    6069.8 36647 2171.0
## + zn       1    5549.7 37167 2178.1
## + black    1    4749.9 37966 2188.9
## + dis      1    2668.2 40048 2215.9
## + chas     1    1312.1 41404 2232.7
```

```
## <none>                 42716 2246.5
##
## Step:  AIC=1851.01
## medv ~ lstat
##
##           Df Sum of Sq   RSS    AIC
## + rm       1    4033.1 15439 1735.6
## + ptratio  1    2670.1 16802 1778.4
## + chas     1     786.3 18686 1832.2
## + dis      1     772.4 18700 1832.5
## + age      1     304.3 19168 1845.0
## + tax      1     274.4 19198 1845.8
## + black    1     198.3 19274 1847.8
## + zn       1     160.3 19312 1848.8
## + crim     1     146.9 19325 1849.2
## + indus    1      98.7 19374 1850.4
## <none>                 19472 1851.0
## + rad      1      25.1 19447 1852.4
## + nox      1       4.8 19468 1852.9
##
## Step:  AIC=1735.58
## medv ~ lstat + rm
##
##           Df Sum of Sq   RSS    AIC
## + ptratio  1   1711.32 13728 1678.1
## + chas     1    548.53 14891 1719.3
## + black    1    512.31 14927 1720.5
## + tax      1    425.16 15014 1723.5
## + dis      1    351.15 15088 1725.9
## + crim     1    311.42 15128 1727.3
## + rad      1    180.45 15259 1731.6
## + indus    1     61.09 15378 1735.6
## <none>                 15439 1735.6
## + zn       1     56.56 15383 1735.7
## + age      1     20.18 15419 1736.9
## + nox      1     14.90 15424 1737.1
##
## Step:  AIC=1678.13
## medv ~ lstat + rm + ptratio
##
##         Df Sum of Sq   RSS    AIC
## + dis    1    499.08 13229 1661.4
## + black  1    389.68 13338 1665.6
## + chas   1    377.96 13350 1666.0
## + crim   1    122.52 13606 1675.6
## + age    1     66.24 13662 1677.7
## <none>               13728 1678.1
## + tax    1     44.36 13684 1678.5
## + nox    1     24.81 13703 1679.2
## + zn     1     14.96 13713 1679.6
## + rad    1      6.07 13722 1679.9
## + indus  1      0.83 13727 1680.1
##
## Step:  AIC=1661.39
```

```
## medv ~ lstat + rm + ptratio + dis
##
##         Df Sum of Sq   RSS    AIC
## + nox    1    759.56 12469 1633.5
## + black  1    502.64 12726 1643.8
## + chas   1    267.43 12962 1653.1
## + indus  1    242.65 12986 1654.0
## + tax    1    240.34 12989 1654.1
## + crim   1    233.54 12995 1654.4
## + zn     1    144.81 13084 1657.8
## + age    1     61.36 13168 1661.0
## <none>               13229 1661.4
## + rad    1     22.40 13206 1662.5
##
## Step:  AIC=1633.47
## medv ~ lstat + rm + ptratio + dis + nox
##
##         Df Sum of Sq   RSS    AIC
## + chas   1    328.27 12141 1622.0
## + black  1    311.83 12158 1622.7
## + zn     1    151.71 12318 1629.3
## + crim   1    141.43 12328 1629.7
## + rad    1     53.48 12416 1633.3
## <none>               12469 1633.5
## + indus  1     17.10 12452 1634.8
## + tax    1     10.50 12459 1635.0
## + age    1      0.25 12469 1635.5
##
## Step:  AIC=1621.97
## medv ~ lstat + rm + ptratio + dis + nox + chas
##
##         Df Sum of Sq   RSS    AIC
## + black  1   272.837 11868 1612.5
## + zn     1   164.406 11977 1617.1
## + crim   1   116.330 12025 1619.1
## + rad    1    58.556 12082 1621.5
## <none>               12141 1622.0
## + indus  1    26.274 12115 1622.9
## + tax    1     4.187 12137 1623.8
## + age    1     2.331 12139 1623.9
##
## Step:  AIC=1612.47
## medv ~ lstat + rm + ptratio + dis + nox + chas + black
##
##         Df Sum of Sq   RSS    AIC
## + zn     1   189.936 11678 1606.3
## + rad    1   144.320 11724 1608.3
## + crim   1    55.633 11813 1612.1
## <none>               11868 1612.5
## + indus  1    15.584 11853 1613.8
## + age    1     9.446 11859 1614.1
## + tax    1     2.703 11866 1614.4
##
## Step:  AIC=1606.31
```

```
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn
##
##         Df Sum of Sq   RSS    AIC
## + crim   1    94.712 11584 1604.2
## + rad    1    93.614 11585 1604.2
## <none>               11678 1606.3
## + indus  1    16.048 11662 1607.6
## + tax    1     3.952 11674 1608.1
## + age    1     1.491 11677 1608.2
##
## Step:  AIC=1604.19
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
##     crim
##
##         Df Sum of Sq   RSS    AIC
## + rad    1   228.604 11355 1596.1
## <none>               11584 1604.2
## + indus  1    15.773 11568 1605.5
## + age    1     2.470 11581 1606.1
## + tax    1     1.305 11582 1606.1
##
## Step:  AIC=1596.1
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
##     crim + rad
##
##         Df Sum of Sq   RSS    AIC
## + tax    1   273.619 11081 1585.8
## <none>               11355 1596.1
## + indus  1    33.894 11321 1596.6
## + age    1     0.096 11355 1598.1
##
## Step:  AIC=1585.76
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
##     crim + rad + tax
##
##         Df Sum of Sq   RSS    AIC
## <none>               11081 1585.8
## + indus  1   2.51754 11079 1587.7
## + age    1   0.06271 11081 1587.8
##
## Call:
## lm(formula = medv ~ lstat + rm + ptratio + dis + nox + chas +
##     black + zn + crim + rad + tax, data = Boston)
##
## Coefficients:
## (Intercept)        lstat           rm      ptratio          dis
##   36.341145    -0.522553     3.801579    -0.946525    -1.492711
##         nox        chas1        black           zn         crim
##  -17.376023     2.718716     0.009291     0.045845    -0.108413
##         rad          tax
##    0.299608    -0.011778
```

Backward elimination:

```r
step(full, direction="backward")
```

```
## Start:  AIC=1589.64
## medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + black + lstat
##
##           Df Sum of Sq   RSS    AIC
## - age      1      0.06 11079 1587.7
## - indus    1      2.52 11081 1587.8
## <none>                 11079 1589.6
## - chas     1    218.97 11298 1597.5
## - tax      1    242.26 11321 1598.6
## - crim     1    243.22 11322 1598.6
## - zn       1    257.49 11336 1599.3
## - black    1    270.63 11349 1599.8
## - rad      1    479.15 11558 1609.1
## - nox      1    487.16 11566 1609.4
## - ptratio  1   1194.23 12273 1639.4
## - dis      1   1232.41 12311 1641.0
## - rm       1   1871.32 12950 1666.6
## - lstat    1   2410.84 13490 1687.3
##
## Step:  AIC=1587.65
## medv ~ crim + zn + indus + chas + nox + rm + dis + rad + tax +
##     ptratio + black + lstat
##
##           Df Sum of Sq   RSS    AIC
## - indus    1      2.52 11081 1585.8
## <none>                 11079 1587.7
## - chas     1    219.91 11299 1595.6
## - tax      1    242.24 11321 1596.6
## - crim     1    243.20 11322 1596.6
## - zn       1    260.32 11339 1597.4
## - black    1    272.26 11351 1597.9
## - rad      1    481.09 11560 1607.2
## - nox      1    520.87 11600 1608.9
## - ptratio  1   1200.23 12279 1637.7
## - dis      1   1352.26 12431 1643.9
## - rm       1   1959.55 13038 1668.0
## - lstat    1   2718.88 13798 1696.7
##
## Step:  AIC=1585.76
## medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
##     black + lstat
##
##           Df Sum of Sq   RSS    AIC
## <none>                 11081 1585.8
## - chas     1    227.21 11309 1594.0
## - crim     1    245.37 11327 1594.8
## - zn       1    257.82 11339 1595.4
## - black    1    270.82 11352 1596.0
## - tax      1    273.62 11355 1596.1
## - rad      1    500.92 11582 1606.1
## - nox      1    541.91 11623 1607.9
```

```
## - ptratio  1   1206.45 12288 1636.0
## - dis      1   1448.94 12530 1645.9
## - rm       1   1963.66 13045 1666.3
## - lstat    1   2723.48 13805 1695.0
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + black + lstat, data = Boston)
##
## Coefficients:
## (Intercept)         crim           zn        chas1          nox
##   36.341145    -0.108413     0.045845     2.718716   -17.376023
##          rm          dis          rad          tax      ptratio
##    3.801579    -1.492711     0.299608    -0.011778    -0.946525
##       black        lstat
##    0.009291    -0.522553
```

Y paso a paso *stepwise regression*:

```
step(null, scope = list(upper=full), data=Boston, direction="both")
```

```
## Start:  AIC=2246.51
## medv ~ 1
##
##           Df Sum of Sq   RSS    AIC
## + lstat    1   23243.9 19472 1851.0
## + rm       1   20654.4 22062 1914.2
## + ptratio  1   11014.3 31702 2097.6
## + indus    1    9995.2 32721 2113.6
## + tax      1    9377.3 33339 2123.1
## + nox      1    7800.1 34916 2146.5
## + crim     1    6440.8 36276 2165.8
## + rad      1    6221.1 36495 2168.9
## + age      1    6069.8 36647 2171.0
## + zn       1    5549.7 37167 2178.1
## + black    1    4749.9 37966 2188.9
## + dis      1    2668.2 40048 2215.9
## + chas     1    1312.1 41404 2232.7
## <none>                 42716 2246.5
##
## Step:  AIC=1851.01
## medv ~ lstat
##
##           Df Sum of Sq   RSS    AIC
## + rm       1    4033.1 15439 1735.6
## + ptratio  1    2670.1 16802 1778.4
## + chas     1     786.3 18686 1832.2
## + dis      1     772.4 18700 1832.5
## + age      1     304.3 19168 1845.0
## + tax      1     274.4 19198 1845.8
## + black    1     198.3 19274 1847.8
## + zn       1     160.3 19312 1848.8
## + crim     1     146.9 19325 1849.2
## + indus    1      98.7 19374 1850.4
## <none>                 19472 1851.0
```

```
## + rad      1      25.1 19447 1852.4
## + nox      1       4.8 19468 1852.9
## - lstat    1   23243.9 42716 2246.5
##
## Step:  AIC=1735.58
## medv ~ lstat + rm
##
##           Df Sum of Sq   RSS    AIC
## + ptratio  1    1711.3 13728 1678.1
## + chas     1     548.5 14891 1719.3
## + black    1     512.3 14927 1720.5
## + tax      1     425.2 15014 1723.5
## + dis      1     351.2 15088 1725.9
## + crim     1     311.4 15128 1727.3
## + rad      1     180.5 15259 1731.6
## + indus    1      61.1 15378 1735.6
## <none>                  15439 1735.6
## + zn       1      56.6 15383 1735.7
## + age      1      20.2 15419 1736.9
## + nox      1      14.9 15424 1737.1
## - rm       1    4033.1 19472 1851.0
## - lstat    1    6622.6 22062 1914.2
##
## Step:  AIC=1678.13
## medv ~ lstat + rm + ptratio
##
##           Df Sum of Sq   RSS    AIC
## + dis      1     499.1 13229 1661.4
## + black    1     389.7 13338 1665.6
## + chas     1     378.0 13350 1666.0
## + crim     1     122.5 13606 1675.6
## + age      1      66.2 13662 1677.7
## <none>                  13728 1678.1
## + tax      1      44.4 13684 1678.5
## + nox      1      24.8 13703 1679.2
## + zn       1      15.0 13713 1679.6
## + rad      1       6.1 13722 1679.9
## + indus    1       0.8 13727 1680.1
## - ptratio  1    1711.3 15439 1735.6
## - rm       1    3074.3 16802 1778.4
## - lstat    1    5013.6 18742 1833.7
##
## Step:  AIC=1661.39
## medv ~ lstat + rm + ptratio + dis
##
##           Df Sum of Sq   RSS    AIC
## + nox      1     759.6 12469 1633.5
## + black    1     502.6 12726 1643.8
## + chas     1     267.4 12962 1653.1
## + indus    1     242.6 12986 1654.0
## + tax      1     240.3 12989 1654.1
## + crim     1     233.5 12995 1654.4
## + zn       1     144.8 13084 1657.8
## + age      1      61.4 13168 1661.0
```

```
## <none>                 13229 1661.4
## + rad      1      22.4 13206 1662.5
## - dis      1     499.1 13728 1678.1
## - ptratio  1    1859.3 15088 1725.9
## - rm       1    2622.6 15852 1750.9
## - lstat    1    5349.2 18578 1831.2
##
## Step:  AIC=1633.47
## medv ~ lstat + rm + ptratio + dis + nox
##
##           Df Sum of Sq   RSS    AIC
## + chas     1     328.3 12141 1622.0
## + black    1     311.8 12158 1622.7
## + zn       1     151.7 12318 1629.3
## + crim     1     141.4 12328 1629.7
## + rad      1      53.5 12416 1633.3
## <none>                 12469 1633.5
## + indus    1      17.1 12452 1634.8
## + tax      1      10.5 12459 1635.0
## + age      1       0.2 12469 1635.5
## - nox      1     759.6 13229 1661.4
## - dis      1    1233.8 13703 1679.2
## - ptratio  1    2116.5 14586 1710.8
## - rm       1    2546.2 15016 1725.5
## - lstat    1    3664.3 16134 1761.8
##
## Step:  AIC=1621.97
## medv ~ lstat + rm + ptratio + dis + nox + chas
##
##           Df Sum of Sq   RSS    AIC
## + black    1     272.8 11868 1612.5
## + zn       1     164.4 11977 1617.1
## + crim     1     116.3 12025 1619.1
## + rad      1      58.6 12082 1621.5
## <none>                 12141 1622.0
## + indus    1      26.3 12115 1622.9
## + tax      1       4.2 12137 1623.8
## + age      1       2.3 12139 1623.9
## - chas     1     328.3 12469 1633.5
## - nox      1     820.4 12962 1653.1
## - dis      1    1146.8 13288 1665.6
## - ptratio  1    1924.9 14066 1694.4
## - rm       1    2480.7 14622 1714.0
## - lstat    1    3509.3 15650 1748.5
##
## Step:  AIC=1612.47
## medv ~ lstat + rm + ptratio + dis + nox + chas + black
##
##           Df Sum of Sq   RSS    AIC
## + zn       1    189.94 11678 1606.3
## + rad      1    144.32 11724 1608.3
## + crim     1     55.63 11813 1612.1
## <none>                 11868 1612.5
## + indus    1     15.58 11853 1613.8
```

```
## + age      1      9.45 11859 1614.1
## + tax      1      2.70 11866 1614.4
## - black    1    272.84 12141 1622.0
## - chas     1    289.27 12158 1622.7
## - nox      1    626.85 12495 1636.5
## - dis      1   1103.33 12972 1655.5
## - ptratio  1   1804.30 13672 1682.1
## - rm       1   2658.21 14526 1712.7
## - lstat    1   2991.55 14860 1724.2
##
## Step:  AIC=1606.31
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn
##
##           Df Sum of Sq   RSS    AIC
## + crim     1     94.71 11584 1604.2
## + rad      1     93.61 11585 1604.2
## <none>                 11678 1606.3
## + indus    1     16.05 11662 1607.6
## + tax      1      3.95 11674 1608.1
## + age      1      1.49 11677 1608.2
## - zn       1    189.94 11868 1612.5
## - black    1    298.37 11977 1617.1
## - chas     1    300.42 11979 1617.2
## - nox      1    627.62 12306 1630.8
## - dis      1   1276.45 12955 1656.8
## - ptratio  1   1364.63 13043 1660.2
## - rm       1   2384.55 14063 1698.3
## - lstat    1   3052.50 14731 1721.8
##
## Step:  AIC=1604.19
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
##     crim
##
##           Df Sum of Sq   RSS    AIC
## + rad      1    228.60 11355 1596.1
## <none>                 11584 1604.2
## + indus    1     15.77 11568 1605.5
## + age      1      2.47 11581 1606.1
## + tax      1      1.31 11582 1606.1
## - crim     1     94.71 11678 1606.3
## - black    1    222.18 11806 1611.8
## - zn       1    229.02 11813 1612.1
## - chas     1    284.34 11868 1614.5
## - nox      1    578.44 12162 1626.8
## - ptratio  1   1192.90 12776 1651.8
## - dis      1   1345.70 12929 1657.8
## - rm       1   2419.57 14003 1698.2
## - lstat    1   2753.42 14337 1710.1
##
## Step:  AIC=1596.1
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
##     crim + rad
##
##           Df Sum of Sq   RSS    AIC
```

```
## + tax      1    273.62 11081 1585.8
## <none>                 11355 1596.1
## + indus    1     33.89 11321 1596.6
## + age      1      0.10 11355 1598.1
## - zn       1    171.14 11526 1601.7
## - rad      1    228.60 11584 1604.2
## - crim     1    229.70 11585 1604.2
## - chas     1    272.67 11628 1606.1
## - black    1    295.78 11651 1607.1
## - nox      1    785.16 12140 1627.9
## - dis      1   1341.37 12696 1650.6
## - ptratio  1   1419.77 12775 1653.7
## - rm       1   2182.57 13538 1683.1
## - lstat    1   2785.28 14140 1705.1
##
## Step:  AIC=1585.76
## medv ~ lstat + rm + ptratio + dis + nox + chas + black + zn +
##     crim + rad + tax
##
##           Df Sum of Sq   RSS    AIC
## <none>                 11081 1585.8
## + indus    1      2.52 11079 1587.7
## + age      1      0.06 11081 1587.8
## - chas     1    227.21 11309 1594.0
## - crim     1    245.37 11327 1594.8
## - zn       1    257.82 11339 1595.4
## - black    1    270.82 11352 1596.0
## - tax      1    273.62 11355 1596.1
## - rad      1    500.92 11582 1606.1
## - nox      1    541.91 11623 1607.9
## - ptratio  1   1206.45 12288 1636.0
## - dis      1   1448.94 12530 1645.9
## - rm       1   1963.66 13045 1666.3
## - lstat    1   2723.48 13805 1695.0
##
## Call:
## lm(formula = medv ~ lstat + rm + ptratio + dis + nox + chas +
##     black + zn + crim + rad + tax, data = Boston)
##
## Coefficients:
## (Intercept)        lstat           rm      ptratio          dis
##   36.341145    -0.522553     3.801579    -0.946525    -1.492711
##         nox        chas1        black           zn         crim
##  -17.376023     2.718716     0.009291     0.045845    -0.108413
##         rad          tax
##    0.299608    -0.011778
```

Regresión Logística

Normalmente se utiliza una regresión logística cuando hay una variable de resultado dicotómica (como ganar/perder, sano/enfermo) y una variable predictora continua que está relacionada con la probabilidad o las probabilidades de la variable de resultado. También puede usarse con predictores categóricos y con

múltiples predictores.

Si usamos una regresión lineal para modelar una variable dicotómica (como Y), el modelo resultante podría no restringir los Y's previstos dentro de 0 y 1. Además, otros supuestos de regresión lineal como la normalidad de errores pueden ser violados. Así que en su lugar, modelamos las probabilidades log del evento $\ln(\frac{p}{1-p})$ o logit, donde, p es la probabilidad del evento.

$$z_i = \ln(\frac{p_i}{1-p_i}) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

La ecuación anterior se puede modelar usando `glm()` con el argumento `family "binomial"`. Pero estamos más interesados en la probabilidad del evento, que en las probabilidades logarítmicas del evento. Por lo tanto, los valores predichos del modelo anterior, es decir, las probabilidades logarítmicas del evento, se pueden convertir en probabilidad de evento como sigue (antilogit):

$$p_i = 1 - \frac{1}{1 + \exp(z_i)}$$

Esta conversión se logra utilizando la función `plogis()`.

Dado que la variable de respuesta es binaria, un modelo de regresión múltiple no es adecuado para un análisis de regresión.

Podemos escribir

$$\mathbb{P}\mathrm{r}(y_i = 1) = \pi_i \qquad \mathbb{P}\mathrm{r}(y_i = 0) = 1 - \pi_i$$

El modelo

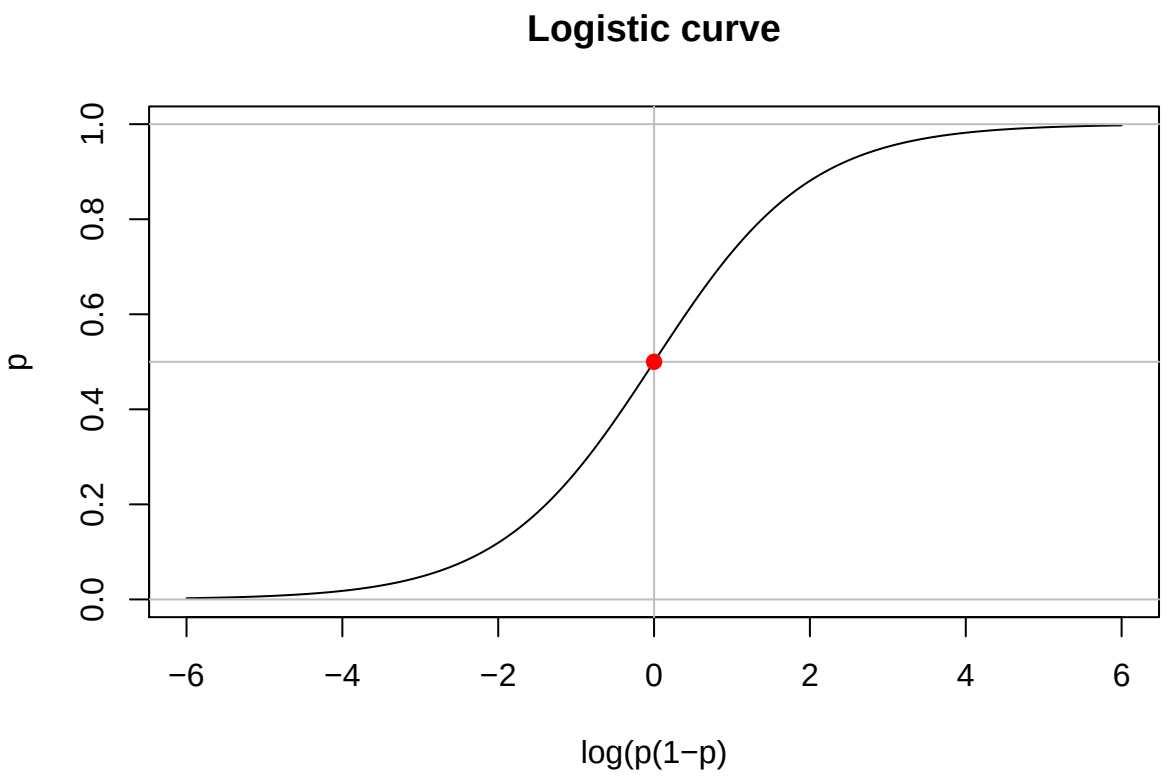
$$\mathrm{logit}(\pi) = \mathrm{logit}\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

El logit de una probabilidad es simplemente el log de las probabilidades de la respuesta tomando el valor una transformación logit o de p: $logit(p) = \log(p/1-p)$. Propiedades

- Si `odds(y=1) = 1`, entonces `logit(p) = 0`.
- Si `odds(y=1) < 1`, entonces `logit(p) < 0`.
- Si `odds(y=1) > 1`, entonces `logit(p) > 0`.

Cuando la respuesta es una variable binaria (dicotómica), y x es numérica, la regresión logística ajusta una curva logística a la relación entre x y y. Por lo tanto, la regresión logística es la regresión lineal en la transformación logit de y, donde y es la proporción (o probabilidad) de éxito en cada valor de x. Sin embargo, se evitar la tentación de hacer una regresión lineal, ya que ni la normalidad ni la suposición de homoscedasticidad se satisfacen.

```
x <- seq(-6,6,0.01)
logistic <- exp(x)/(1+exp(x))
plot(x,logistic,t='l',main="Logistic curve",ylab="p",xlab="log(p(1-p)")
abline(h=c(0,0.5,1),v=0,col="grey")
points(0,0.5,pch=19,col=2)
```



German credit Data

Cuando un banco recibe una solicitud de préstamo, basado en el perfil del solicitante, el banco tiene que tomar una decisión sobre si seguir adelante con la aprobación del préstamo o no. Dos tipos de riesgos están asociados con la decisión del banco -

- Si el solicitante tiene un buen riesgo de crédito, es decir, es probable que pague el préstamo, entonces no aprobar el préstamo a la persona resulta en una pérdida de negocio para el banco.
- Si el solicitante tiene un riesgo de crédito malo, es decir, no es probable que pague el préstamo, entonces aprobar el préstamo a la persona resulta en una pérdida financiera para el banco.

Minimización del riesgo y maximización de la rentabilidad por parte del banco.

Para minimizar las pérdidas desde la perspectiva del banco, el banco necesita una regla de decisión con respecto a a quién dar la aprobación del préstamo y a quién no. Los perfiles demográficos y socioeconómicos de un solicitante son considerados por los administradores de préstamos antes de que se tome una decisión con respecto a su solicitud de préstamo.

Los datos de crédito alemanes contienen datos sobre 20 variables y la clasificación de si un solicitante se considera un riesgo de crédito `Bueno` o `Malo` para 1000 solicitantes de préstamo. Se espera que un modelo predictivo desarrollado a partir de estos datos proporcione una guía al gerente del banco para tomar la decisión de aprobar un préstamo a un posible solicitante basándose en sus perfiles.

```
data <- read.table("http://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/german/german.data", h

colnames(data)<-c("account.status","months",
                  "credit.history","purpose","credit.amount",
                  "savings","employment","installment.rate","personal.status",
                  "guarantors","residence","property","age","other.installments",
                  "housing","credit.cards","job","dependents","phone","foreign.worker","credit.rating")
```

```
head(data)
```

```
##   account.status months credit.history purpose credit.amount savings
## 1            A11      6            A34     A43          1169     A65
## 2            A12     48            A32     A43          5951     A61
## 3            A14     12            A34     A46          2096     A61
## 4            A11     42            A32     A42          7882     A61
## 5            A11     24            A33     A40          4870     A61
## 6            A14     36            A32     A46          9055     A65
##   employment installment.rate personal.status guarantors residence
## 1        A75                4            A93       A101         4
## 2        A73                2            A92       A101         2
## 3        A74                2            A93       A101         3
## 4        A74                2            A93       A103         4
## 5        A73                3            A93       A101         4
## 6        A73                2            A93       A101         4
##   property age other.installments housing credit.cards  job dependents
## 1     A121  67               A143    A152            2 A173          1
## 2     A121  22               A143    A152            1 A173          1
## 3     A121  49               A143    A152            1 A172          2
## 4     A122  45               A143    A153            1 A173          2
## 5     A124  53               A143    A153            2 A173          2
## 6     A124  35               A143    A153            1 A172          2
##   phone foreign.worker credit.rating
## 1  A192           A201             1
## 2  A191           A201             2
## 3  A191           A201             1
## 4  A191           A201             1
## 5  A191           A201             2
## 6  A192           A201             1
```

```
levels(data$account.status) <- c("<0DM","<200DM",">200DM","NoStatus")
levels(data$credit.history) <- c("No","Allpaid","Allpaidtillnow","Delayinpaying","Critical")
levels(data$purpose) <- c("car(new)","car(used)","furniture/equipment","radio/television","domesticappli
"repairs","education","vacation-doesnotexist?","retraining","business","others")
```

Vamos a dividir el conjunto de datos en 0.7: 0.3 para el entrenamiento y la prueba del modelo. Para la regresión logística, también necesitamos transformar el marco de datos con factores en la matriz con valor biométrico.

```
mat1 <- model.matrix(credit.rating ~ . , data = data)
n<- dim(data)[1]

set.seed(1234)
train<- sample(1:n , 0.7*n)
xtrain<- mat1[train,]
xtest<- mat1[-train,]

ytrain<- data$credit.rating[train]
ytrain <- as.factor(ytrain-1) # convert to 0/1 factor
 ytest<- data$credit.rating[-train]
ytest  <- as.factor(ytest-1) #  convert to 0/1 factor
```

Build the logistic Regression model

```
m1 <- glm(credit.rating ~ . , family = binomial, data= data.frame(credit.rating= ytrain, xtrain))
```

Key Variables for the regression model.

```
sig.var<- summary(m1)$coeff[-1,4] <0.01
names(sig.var)[sig.var == T]
```

```
## [1] "account.statusNoStatus"    "months"
## [3] "credit.historyCritical"    "purposecar.used."
## [5] "purposeradio.television"   "purposedomesticappliances"
## [7] "savingsA64"                "savingsA65"
## [9] "installment.rate"
```

Predict outcome with Logistic Regression model, then use the test dataset to evaluate the model.

```
pred1<- predict.glm(m1,newdata = data.frame(ytest,xtest), type="response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
result1<- table(ytest, floor(pred1+1.5))
result1
```

```
##
## ytest   1   2
##     0 176  25
##     1  51  48
```

```
error1<- sum(result1[1,2], result1[2,1])/sum(result1)
error1
```

```
## [1] 0.2533333
```

Curva ROC con el paquete `ROCR`

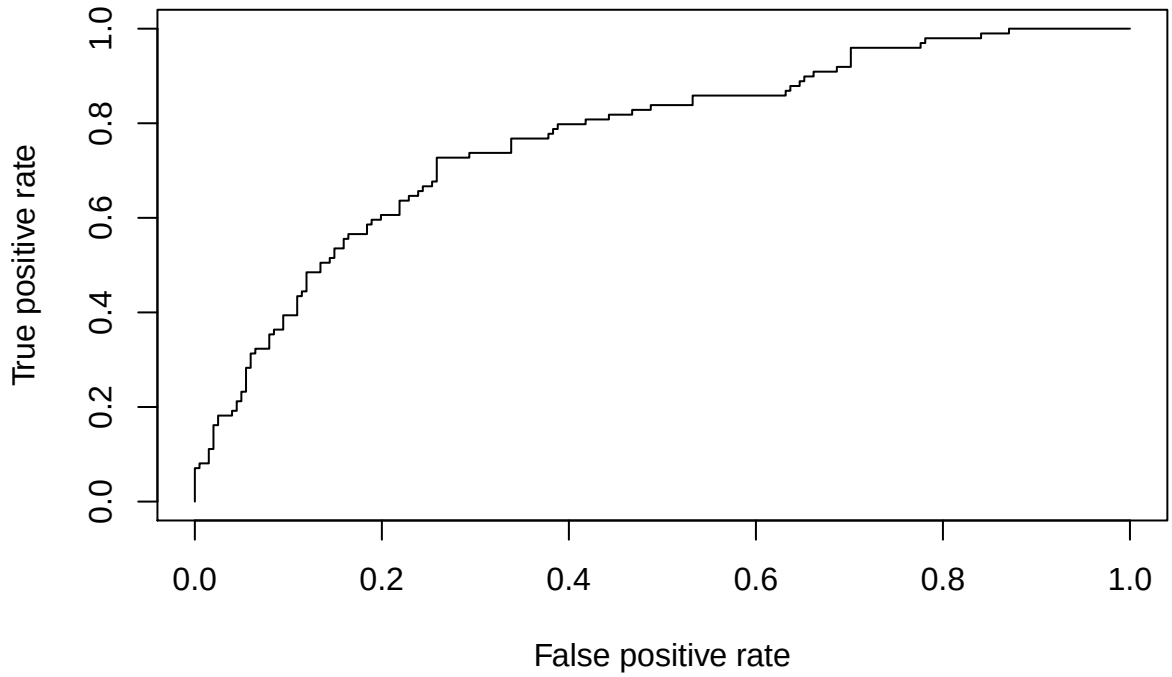
```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred = prediction(pred1,ytest)
perf <- performance(pred, "tpr", "fpr")
plot(perf)
```




```
AUCLog1=performance(pred, measure = "auc")@y.values[[1]]
cat("AUC: ",AUCLog1,"n")
```

```
## AUC:  0.7699382 n
```

Ejemplo

Supongamos el fichero de datos `adult.csv` disponible aquí

Vamos a tratar de predecir la variable respuesta `ABOVE50k` (sueldo >50k) a través de una regresión logistica en base a variables explicativas demográficas.

```
inputData <- read.csv("http://idaejin.github.io/bcam-courses/R/datahack/Modulo1/data/adult.csv")
names(inputData)
```

```
##  [1] "AGE"           "WORKCLASS"     "FNLWGT"        "EDUCATION"
##  [5] "EDUCATIONNUM"  "MARITALSTATUS" "OCCUPATION"    "RELATIONSHIP"
##  [9] "RACE"          "SEX"           "CAPITALGAIN"   "CAPITALLOSS"
## [13] "HOURSPERWEEK"  "NATIVECOUNTRY" "ABOVE50K"
```

Sesgo de clase

Idealmente, la proporción de eventos y no eventos en la variable Y debe ser aproximadamente la misma. Por lo tanto, primero vamos a comprobar la proporción de clases en la variable dependiente `ABOVE50K`.

Claramente, hay un *sesgo de clase*, una condición observada cuando la proporción de eventos es mucho menor que la proporción de no-eventos. Por lo tanto, debemos muestrear las observaciones en proporciones aproximadamente iguales para obtener mejores modelos.

```
table(inputData$ABOVE50K)
```

```
##
##     0     1
```

```
## 24720  7841
```

Crear muestra de entrenamiento y de validación (o test)

Una forma de abordar el problema del sesgo de clase es dibujar los 0 y 1 para el trainingData (muestra de desarrollo) en proporciones iguales. Al hacerlo, pondremos el resto del inputData no incluido en el entrenamiento en testData (muestra de validación). Como resultado, el tamaño de la muestra de desarrollo será menor que la validación, lo que está bien, porque, hay un gran número de observaciones (> 10K).

```
logitMod <- glm(ABOVE50K ~ RELATIONSHIP + AGE + CAPITALGAIN
                + OCCUPATION + EDUCATIONNUM, data=trainingData,
                family=binomial(link="logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
predicted <- plogis(predict(logitMod, testData))  # predicted scores or
# or
# predicted <- predict(logitMod, testData,type="response")  # predicted scores
```

Cuando usamos la función `predict()` en este modelo, se predice el `log(odds)` de la variable Y ($log(\frac{1}{1-p})$). Esto no es lo que queremos en última instancia porque, los valores predichos pueden no estar dentro del rango 0 y 1 como se esperaba. Por lo tanto, para convertirlo en puntajes de probabilidad de predicción que están enlazados entre 0 y 1, usamos el `plogis()` o *type="response"* como argumento de `predict()`.

Decidir el límite óptimo de probabilidad de predicción para el modelo

La puntuación de probabilidad de predicción de corte por defecto es `0,5` o la proporción de 1 y 0 en los datos de entrenamiento. Pero a veces, afinar el límite de probabilidad puede mejorar la precisión tanto en el desarrollo como en las muestras de validación. La función del paquete `InformationValue` llamada `optimalCutoff` proporciona maneras de encontrar el punto de corte óptimo para mejorar la predicción de 1's, 0's, tanto 1 como 0's y o reducir el error de clasificación errónea.

```
library(InformationValue)
optCutOff <- optimalCutoff(testData$ABOVE50K, predicted)[1]
optCutOff
```

```
## [1] 0.89
```

Error de clasificación errónea

El error de clasificación errónea es el porcentaje de desajuste de los valores reales predichos, independientemente de los 1 o los 0. Cuanto menor sea el error de clasificación errónea, mejor será su modelo.

```
misClassError(testData$ABOVE50K, predicted, threshold = optCutOff)
```

```
## [1] 0.0892
```

Curva ROC

La curva ROC (Receiver Operating Characteristics) es una representación gráfica de la sensibilidad frente a la especificidad para un sistema clasificador binario según se varía el umbral de discriminación.

Otra interpretación de este gráfico es la representación de la razón o ratio de verdaderos positivos (VPR = Razón de Verdaderos Positivos) frente a la razón o ratio de falsos positivos (FPR = Razón de Falsos

Positivos) también según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo).

Proporciona herramientas para seleccionar los modelos posiblemente óptimos y descartar modelos subóptimos independientemente de (y antes de especificar) el coste de la distribución de las dos clases sobre las que se decide.

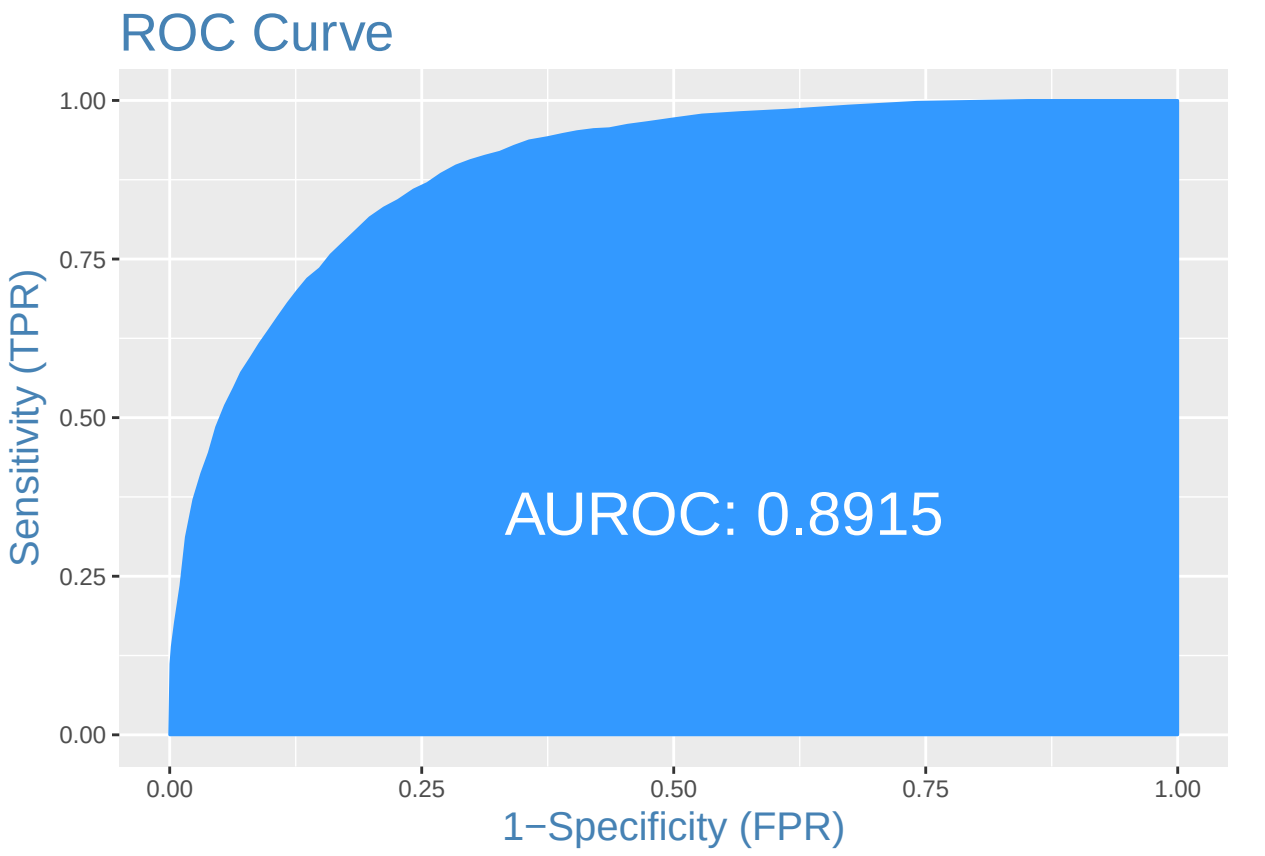
- Verdaderos Positivos (VP) o también éxitos
- Verdaderos Negativos (VN) o también rechazos correctos
- Falsos Positivos (FP) o también falsas alarmas o Error tipo I
- Falsos Negativos (FN) o también, Error de tipo II
- Sensibilidad o Razón de Verdaderos Positivos (VPR) o también razón de éxitos y, recuerdo en recuperación de información,

$$VPR = VP/P = VP/(VP + FN)$$

- Especificidad o Razón de Verdaderos Negativos

$$\text{ESPECIFICIDAD} = VN/N = VN/(FP + VN) = 1 - FPR$$

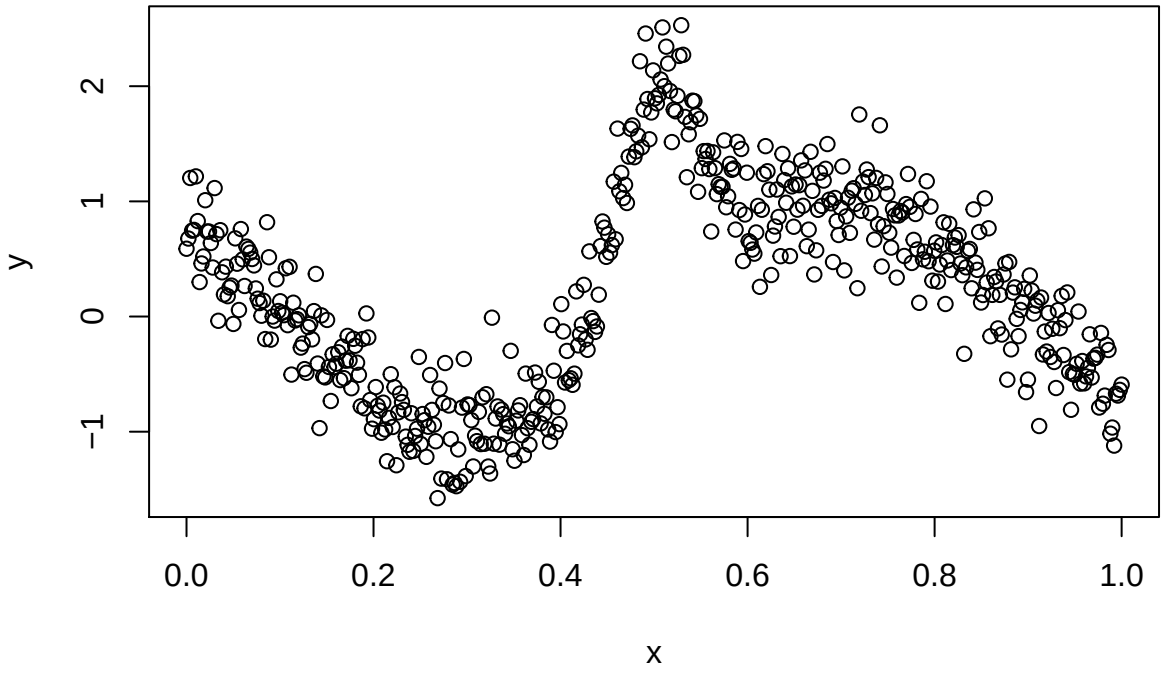
```
plotROC(testData$ABOVE50K, predicted)
```



Modelos Aditivos Generalizados

Un modelo aditivo generalizado (GAM) es un modelo lineal generalizado (GLM) en el que el predictor lineal está dado por una suma especificada por el usuario de funciones suaves de las covariables más una componente paramétrica convencional del predictor lineal.

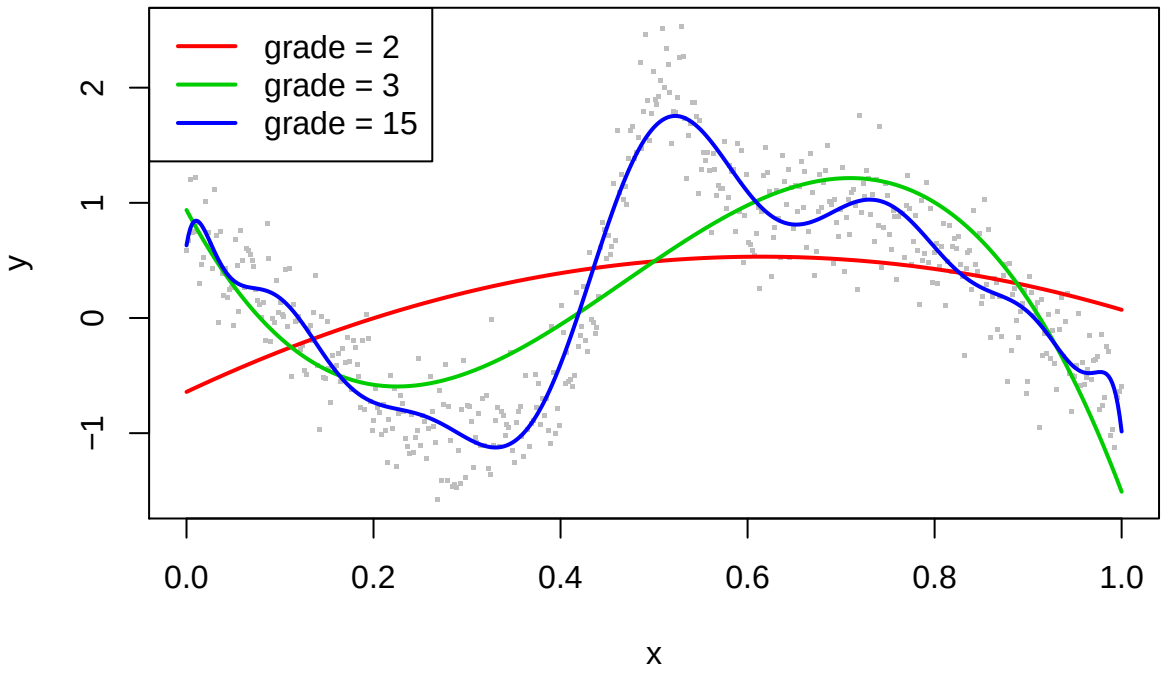
```
set.seed(123)
x = seq(0,1,l=500)
mu = sin(2*(4*x-2)) + 2*exp(-(16^2)*((x-.5)^2))
y = rnorm(500, mu, .3)
d = data.frame(x,y)
plot(d)
```



La regresión polinómica es problemática

Una regresión lineal estándar definitivamente no va a capturar esta relación. Como en el caso anterior, podríamos intentar utilizar la regresión polinómica aquí, por ejemplo, ajustando una función cuadrática o cúbica dentro del marco de regresión estándar. Sin embargo, esto es poco realista en el mejor de los casos y en el peor no es útil para relaciones complejas. A continuación, incluso con un polinomio de grado 15, el ajuste es bastante pobre en muchas áreas, y 'se mueve' en algunos lugares donde no parece haber necesidad de hacerlo.

```
plot(x,y, col="grey", pch=15,cex=.33)
lines(x,fitted(lm(y~poly(x,2))),col=2,lwd=2)
lines(x,fitted(lm(y~poly(x,3))),col=3,lwd=2)
lines(x,fitted(lm(y~poly(x,15))),col=4,lwd=2)
legend("topleft",c("grade = 2","grade = 3","grade = 15"),col=2:4,lty=1,lwd=2)
```



En esencia, un GAM es un GLM. Lo que lo distingue de los que usted conoce es que, a diferencia de un GLM estándar, se compone de una suma de funciones suaves de covariables en lugar de o además de los efectos de covariables lineales estándar. Considere el estándar (g)lm

Para el GAM, podemos especificarlo generalmente de la siguiente manera:

$$y = f(x) + \epsilon,$$

Ahora se trata de alguna función específica (aditiva) de las entradas, que no requerirá que la y (posiblemente transformada) sea una función lineal de x. Se trata de elegir una base, lo que en términos técnicos significa elegir un espacio de funciones para el que f es algún elemento de ella.

Desde el punto de vista práctico, es un medio para captar las relaciones no lineales. Como veremos más adelante, un ejemplo sería elegir un spline cúbico como base. Elegir una base también significa que estamos seleccionando funciones de base, que en realidad se incluirán en el análisis. Podemos añadir más detalles de la siguiente manera:

$$y = f(x) + \epsilon = \sum_{j=1}^{d} B_j(x)\alpha_j + \epsilon$$

cada B_j es una función base que es la x transformada dependiendo del tipo de base considerada, y los α son los coeficientes de regresión correspondientes. Esto puede sonar complicado, hasta que te des cuenta de que lo has hecho antes. Volvamos al polinomio cuadrático, que usa la base polinómica.

$$f(x) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + ... + +\alpha_d x_d$$

En ese caso, $d = 2$ y tenemos nuestra regresión estándar con un término cuadrático, pero de hecho, podemos usar este enfoque para producir las bases para cualquier polinomio.

```
pisa <- read.csv("https://bit.ly/2sMIlAv")

library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-13. For overview type 'help("mgcv-package")'.
```

```
mod.gam.0 <- gam(Overall ~ Income, data = pisa)
summary(mod.gam.0)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ Income
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   204.32      35.37   5.777 4.32e-07 ***
## Income        355.85      46.79   7.606 5.36e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.518   Deviance explained = 52.7%
## GCV = 1504.5  Scale est. = 1448.8    n = 54
```

Lo interesante es que gam permite introducir términos especiales para modelar efectos no lineales. Por ejemplo, en estos datos, los ingresos. El gráfico generado ilustra el impacto (no lineal) del ingreso sobre la variable objetivo.

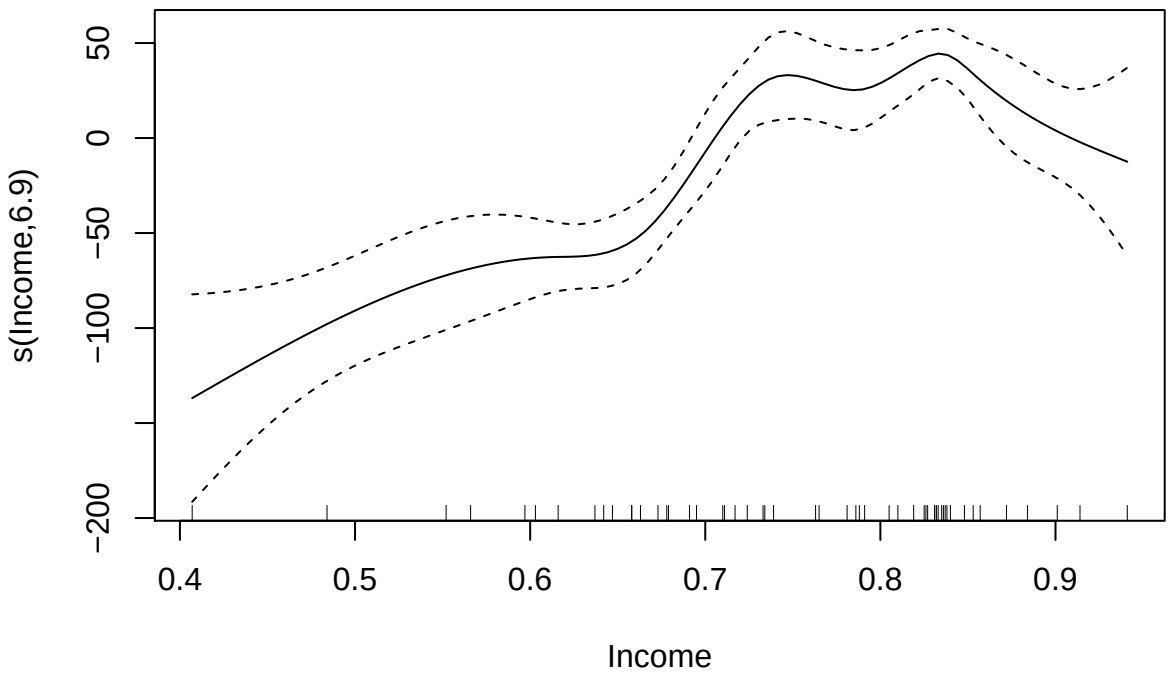
```
mod_gam1 <- gam(Overall ~ s(Income, bs="cr"), data=pisa)
summary(mod_gam1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income, bs = "cr")
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  470.444      4.082   115.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F  p-value
## s(Income) 6.895  7.741 16.67 1.59e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =    0.7   Deviance explained = 73.9%
## GCV = 1053.7  Scale est. = 899.67    n = 54
```

Lo primero a tener en cuenta es que, aparte de la parte suave, nuestro código de modelo es similar a lo que estamos acostumbrados con las funciones principales de R como `lm` y `glm`.

En el resumen, primero vemos la distribución asumida, así como la función de enlace utilizada, en este caso normal e identidad. Después vemos que la salida se separa en partes paramétricas y suaves, o no paramétricas.

```
plot(mod_gam1)
```



GAM multiple

```
mod_lm2 <- gam(Overall ~ Income + Edu + Health, data=pisa)
summary(mod_lm2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ Income + Edu + Health
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   121.18      78.97   1.535   0.1314
## Income        182.32      85.27   2.138   0.0376 *
## Edu           234.11      54.78   4.274 9.06e-05 ***
## Health         27.01     134.90   0.200   0.8421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.616   Deviance explained = 63.9%
## GCV = 1212.3  Scale est. = 1119      n = 52
```

```
mod_gam2 <- gam(Overall ~ s(Income) + s(Edu) + s(Health), data=pisa)
summary(mod_gam2)
```

```
##
## Family: gaussian
```

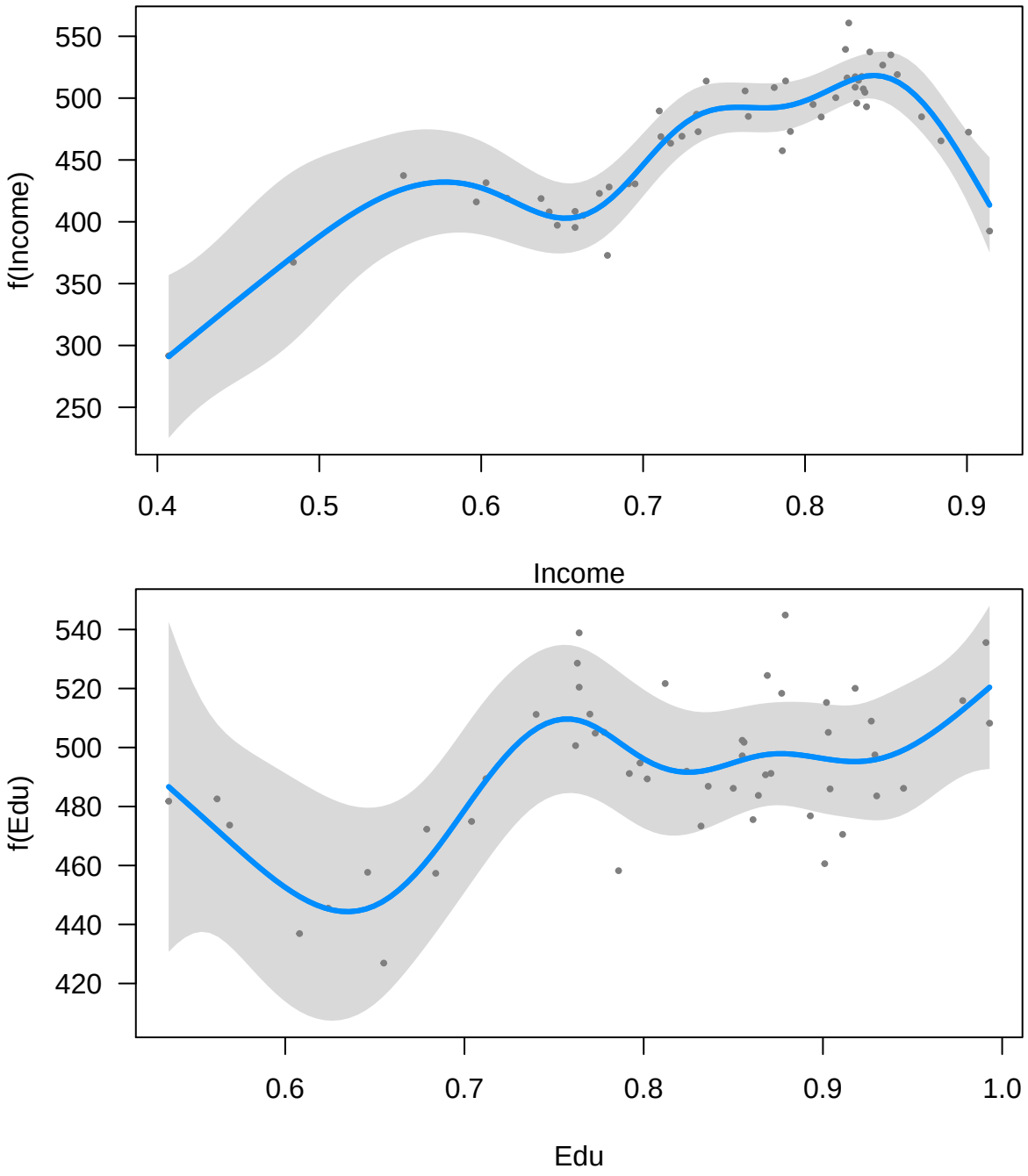
```
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu) + s(Health)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  471.154      2.772     170   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F  p-value
## s(Income) 7.593  8.415 8.826 1.33e-07 ***
## s(Edu)    6.204  7.178 3.308  0.00733 **
## s(Health) 1.000  1.000 2.736  0.10661
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.863   Deviance explained = 90.3%
## GCV = 573.83  Scale est. = 399.5     n = 52
```

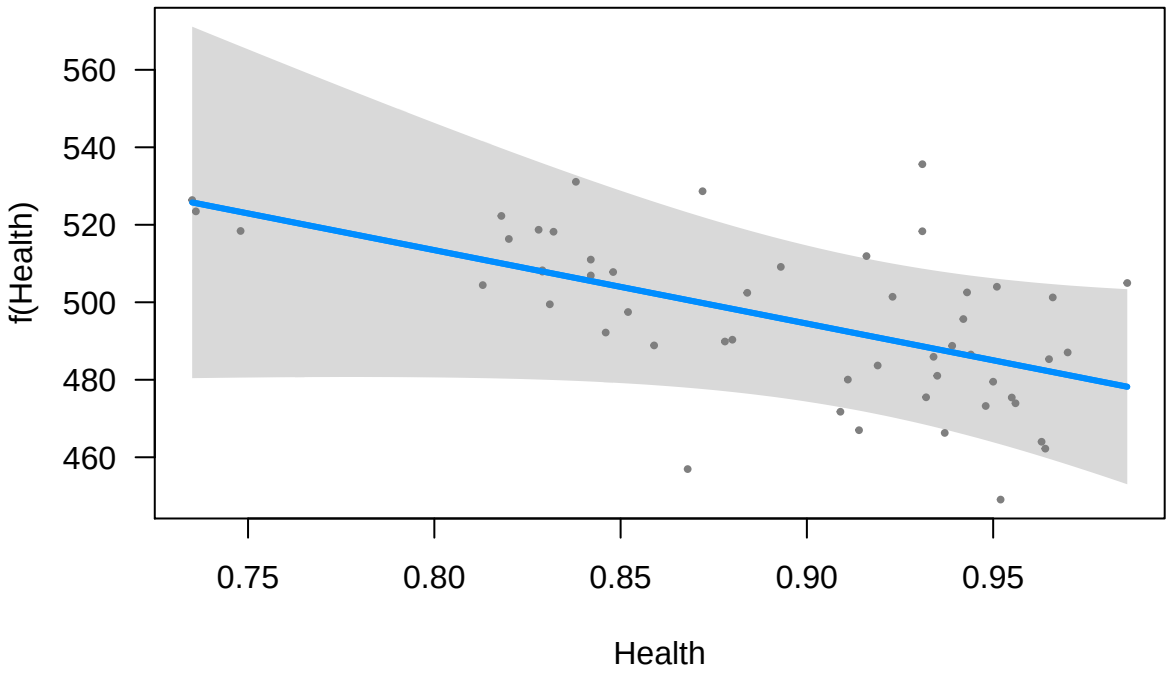
Hay de nuevo un par de cosas de las que hay que tomar nota. En primer lugar, estadísticamente hablando, llegamos a la misma conclusión que el modelo lineal con respecto a los efectos individuales. Hay que tener especialmente en cuenta el efecto del índice de salud. Los grados efectivos de libertad con el valor 1 sugieren que se ha reducido esencialmente a un simple efecto lineal. A continuación se actualizará el modelo para modelar explícitamente el efecto como tal, los resultados son idénticos.

```
mod_gam2B = update(mod_gam2, .~.-s(Health) + Health)
summary(mod_gam2B)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Overall ~ s(Income) + s(Edu) + Health
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    640.3      102.3   6.260 3.06e-07 ***
## Health        -189.5      114.6  -1.654    0.107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df     F  p-value
## s(Income) 7.593  8.415 8.826 1.33e-07 ***
## s(Edu)    6.204  7.178 3.308  0.00733 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.863   Deviance explained = 90.3%
## GCV = 573.83  Scale est. = 399.5     n = 52
```

```
visreg(mod_gam2)
```

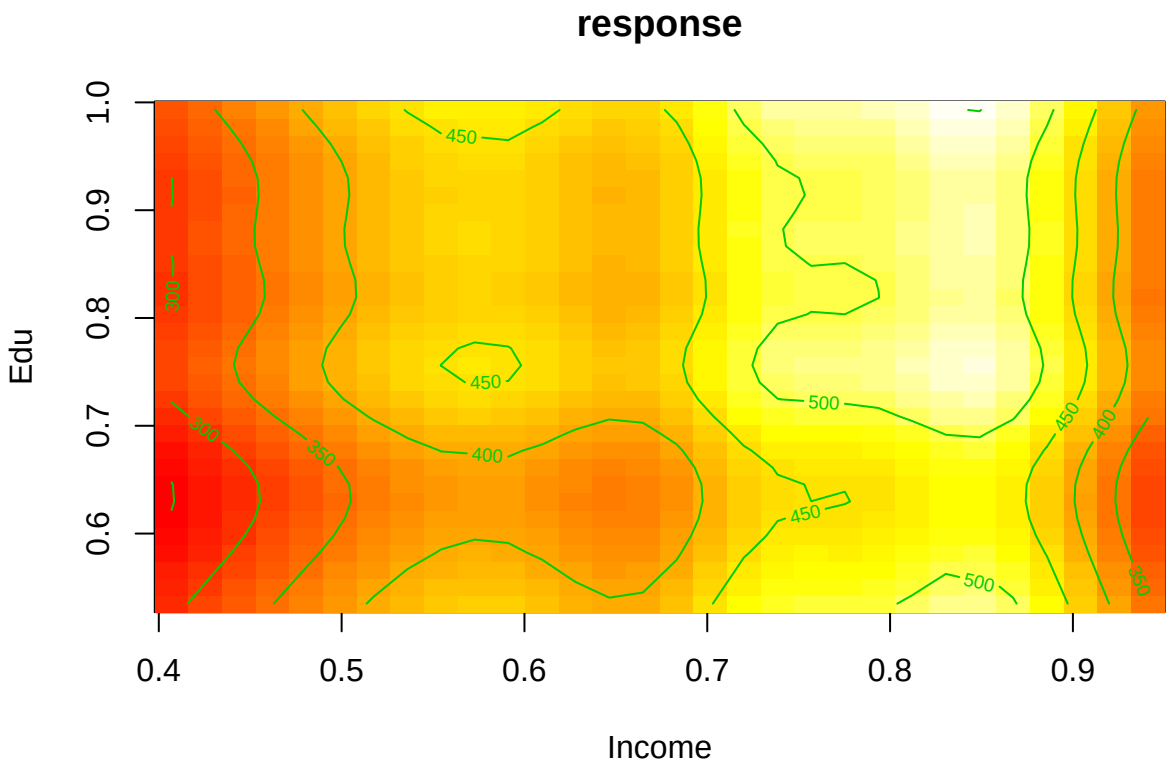




El GAM nos da un sentido para un predictor, pero ahora vamos a echar un vistazo a los ingresos y la educación al mismo tiempo.`vis.gam`, nos permite visualizar superficies en 2d.

Tiene una versión correspondiente de `visreg` que se verá un poco mejor por defecto, visreg2d. El siguiente código producirá un gráfico de contorno con `Ingresos` en el eje x, `Educación` en el eje y, con valores en la escala de respuesta dados por los contornos, con un color más claro que indica valores más altos. El gráfico real que se proporciona en su lugar representa un mapa de calor.

```
vis.gam(mod_gam2, type='response', plot.type='contour')
```



```
visreg2d(mod_gam2, xvar='Income', yvar='Edu', scale='response')
```