

Aprendizaje supervisado con R

Olivier Nuñez

2019-02-27

Índice

1. Introducción	1
Algunos ejemplos	1
Planteamiento del problema	1
Organización del curso	4
Referencia	4
2. Regresión lineal	4
2.1. Regresión lineal simple	4
2.2. Regresión lineal múltiple	9
2.3. Aplicaciones del modelo de regresión múltiple	10
2.4. Selección de los predictores	20
2.5. Predicción	24
3. Clasificación	26
3.1. Planteamiento del problema	26
3.2. Un simple Clasificador	28
3.3. Medida de error en Clasificación	29
3.4. Regresión logística	31
3.5. Análisis Lineal Discriminante	36
3.6. K-vecinos más cercanos	40
4. Regularización	43
4.1. Preámbulo	43
4.2. Regularización con el paquete glmnet	44

1. Introducción

Algunos ejemplos

- Detector de spam
- Riesgo de crédito
- Predicción de los picos de ozono.
- Ayuda para el diagnóstico médico (cáncer de mama)
- Asistencia a la conducción
- Motores de recomendación
- etc ...

Planteamiento del problema

Estamos interesados en controlar/predicir una variable respuesta Y mediante un conjunto de predictores $X = \{X_1, X_2, \dots, X_p\}$, asumiendo que existe una relación entre Y e X del tipo:

$$Y = f(X) + \varepsilon$$

donde ε son las variaciones de Y que no pueden ser descritas por X

El **aprendizaje supervisado** es el conjunto de herramientas estadísticas cuyo objeto es estimar la función desconocida f .

Porqué estimar f ?

Hay dos tipos de razones que motivan la estimación de f :

- Generar conocimiento (Inferencia); f ha de ser interpretable
- Predicción/Clasificación; f puede ser compleja (caja negra)

Cómo estimar f ?

C.J.C. Burges: “A machine with too much capacity is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything seen before; a machine with little capacity is like the botanist lazy brother, who declares that if it is green, it is a tree. Neither can generalize well”.

1.0.0.1. Métodos paramétricos y no-paramétricos

Métodos paramétricos:

Asume una forma particular para f que depende de parámetros que se “aprenden” mediante los datos.

Ejemplo: Modelo de Regresión lineal : (estimación por Mínimos Cuadrados; predicción estable pero sesgada)

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_{01} + \hat{\beta}_2 x_{02} + \dots + \hat{\beta}_k x_{0p}$$

Métodos no-paramétrico:

Utiliza un enfoque algorítmico y local para estimar f

Ejemplo: Vecinos más Cercanos (método flexible pero poco estable, porque depende mucho de la posición de los puntos)

$$\hat{f}(x_0) = \frac{1}{k} \sum_{x_i \in N_k(x_0)} y_i$$

donde $N_k(x)$ corresponde al conjunto de k vecinos de x en la muestra.

1.0.0.2. Cómo seleccionar el mejor modelo

- El modelo de estimación de f ha de tener una capacidad de adaptación a situaciones nuevas.
- Por esa razón, se evalúa el modelo en nuevas observaciones (x_0)

1.0.0.2.1. Métrica para comparar modelos:

Error cuadrático medio (MSE):

$$\begin{aligned} \text{MSE} &= \text{Sesgo}^2 + \text{Varianza} \\ &= E[f(x_0) - \hat{f}(x_0)]^2 + \text{var}[\hat{f}(x_0)] \end{aligned}$$

- **Sesgo:** diferencia media entre la predicción y la verdad

- **Varianza:** Variabilidad de la predicción de una muestra a otra

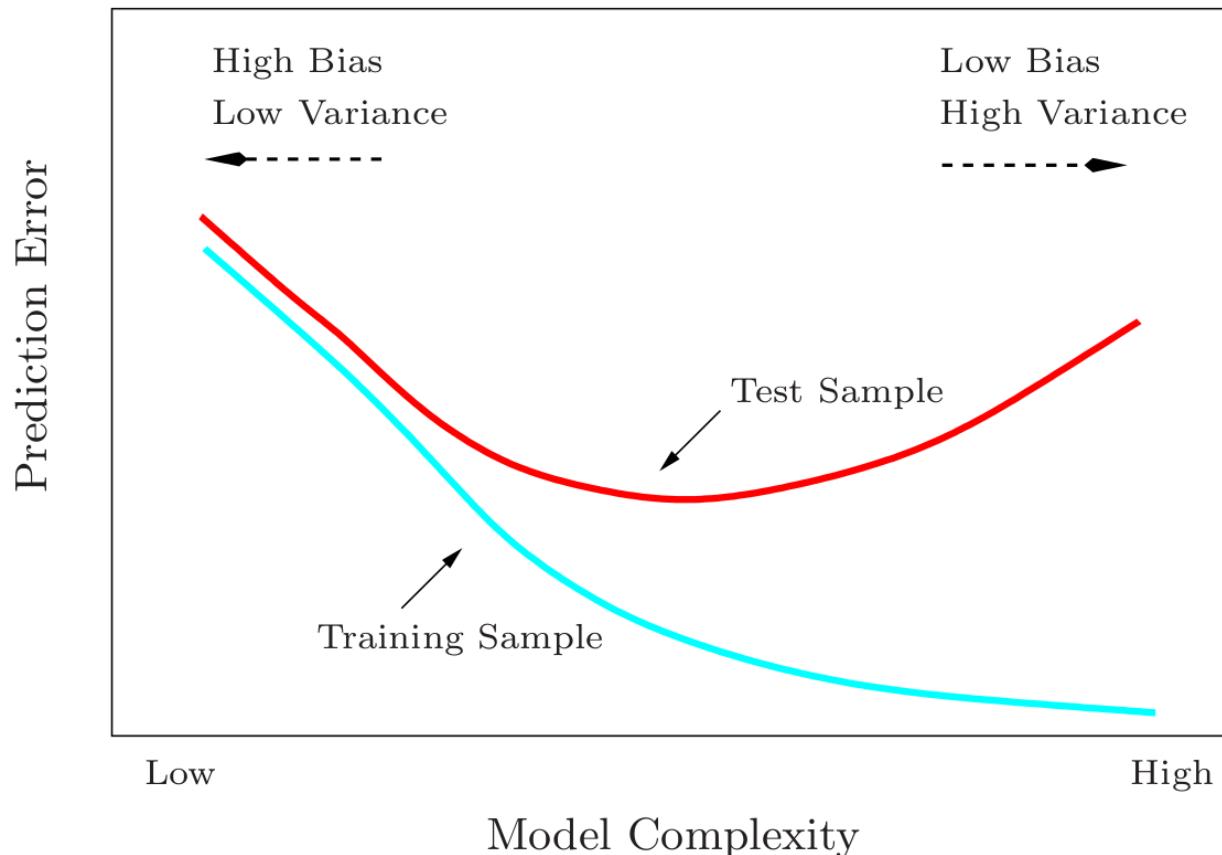


Figura 1:

1.0.0.2.2. Bálance entre sesgo y varianza:

- A medida que aumenta la complejidad (o flexibilidad) del modelo, el sesgo disminuye.
- A medida que aumenta la complejidad del modelo, aumenta la varianza.
- Hay un modelo en medio que da la mejor precisión: minimiza el MSE.

1.0.0.2.3. Evaluar la MSE:

- **Método directo:** se evalúa el modelo en una muestra de validación (*muestra test*) distinta de la muestra utilizada para adiestrar el modelo (*train sample*). Se puede recurrir a la *Validación cruzada* que subdivide de manera iterativa la muestra original en una muestra de entrenamiento y una muestra de validación.
- **Método indirecto:** se approxima la MSE penalizando la bondad/calidad del ajuste del modelo por su complejidad (AIC: criterio de información de Akaike).

1.0.0.3. Limitaciones

- *Maldición de la dimensionalidad* : A medida que crece el número de predictores, (i.e. a medida que crece p), los “vecindarios” deben ser mucho más grandes para contener a “vecinos”, por lo que los métodos locales no son tan locales.
- *No-Free-Lunch Theorem*: No hay un modelo/goritmo de predicción universal, es decir, que sea el mejor en todos los problemas.

Organización del curso

Este curso es una introducción al aprendizaje supervisado en el entorno R y viene organizado de la siguiente manera:

- **Capítulo 2:** Regresión lineal
- **Capítulo 3:** Clasificación
- **Capítulo 4:** Regularización

Referencia

Este curso sobre aprendizaje supervisado está esencialmente basado en el libro “An Introduction to Statistical Learning” de James, Witten, Hastie y Tibshirani (www.StatLearning.com)

A lo largo del curso se utilizará los siguientes paquetes de R:

```
require(tidyverse) #paquete para manejo de datos
require(ISLR) #proporciona las bases de datos incluidas del anterior libro
require(openintro) #proporciona las bases de datos incluidos en el libro "openIntro"
require(glmnet) #paquete para regularización
require(caret)
```

2. Regresión lineal

2.1. Regresión lineal simple

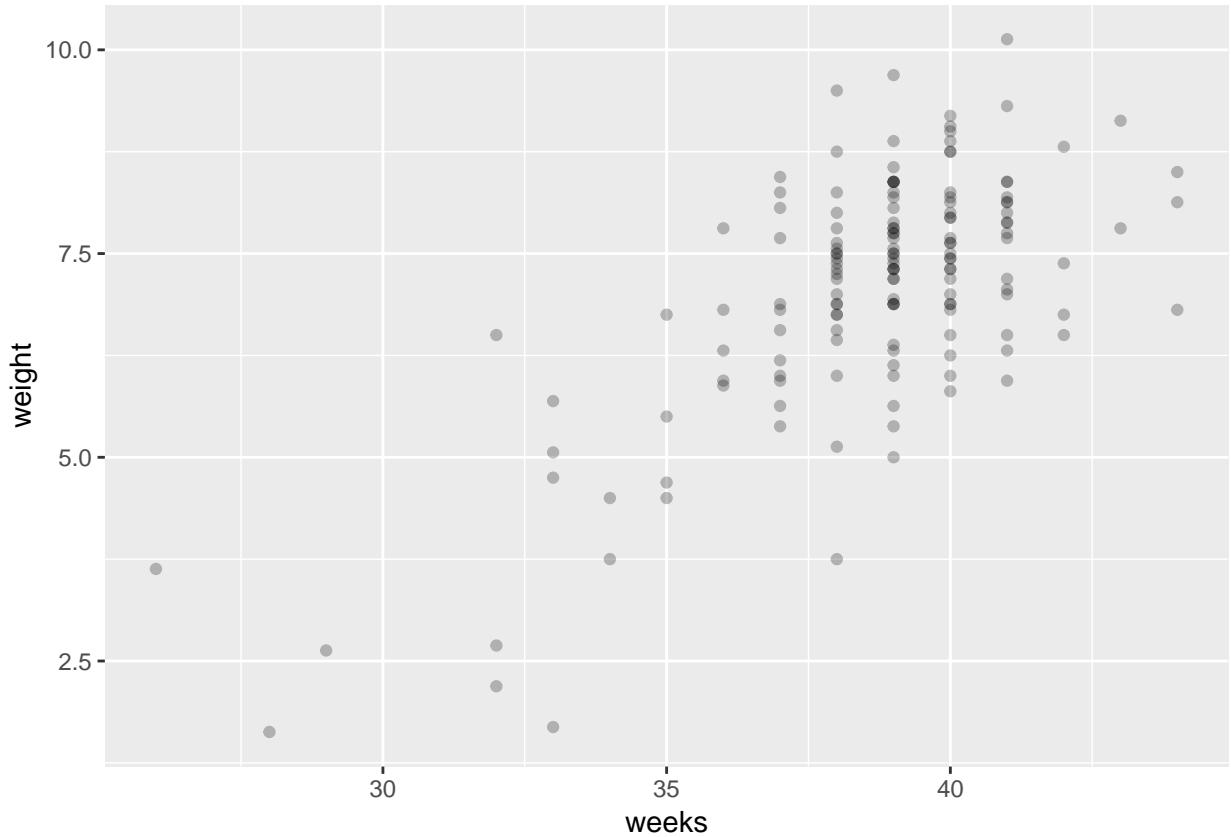
Las técnicas de regresión permiten evaluar la relación entre una variable respuesta Y y un potencial predictor X de esta respuesta. La regresión lineal aproxima esta relación mediante una asociación lineal:

$$Y = \beta_0 + \beta_1 X + \varepsilon,$$

donde ε se interpreta como el error del ajuste lineal y se asume que tiene media cero, varianza constante y un comportamiento normal: $\varepsilon \sim N(0, \sigma)$.

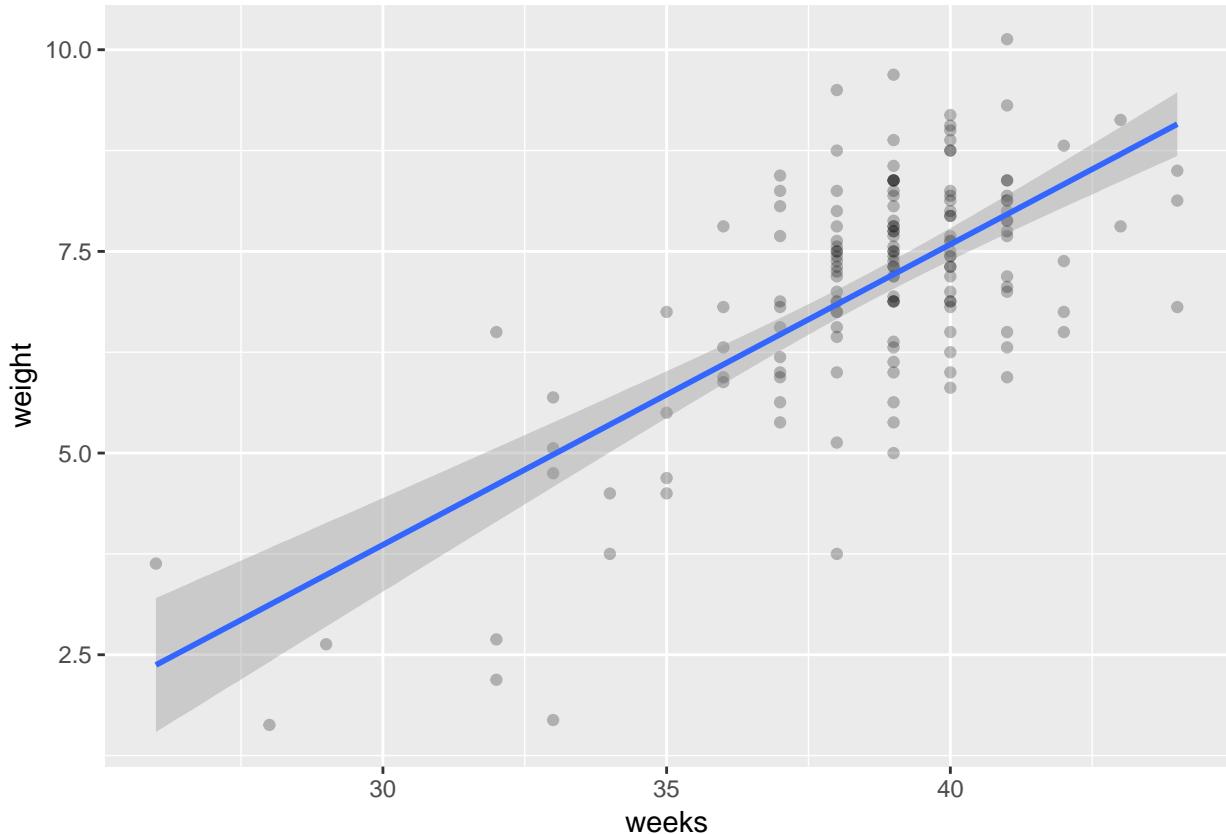
En el ejemplo que se muestra a continuación, se representa el peso al nacer y tiempo de gestación de los neonatos de la base `births`:

```
qplot(weeks, weight, data=births, alpha=I(.25))
```



En este gráfico se puede apreciar una tendencia creciente del peso de los neonatos a medida que aumenta su tiempo de gestación. Esta tendencia se puede estimar ajustando una recta a esta nube de puntos:

```
qplot(weeks,weight,data=births,alpha=I(.25))+geom_smooth(method="lm")
```



En el anterior gráfico, Se puede apreciar como el intervalo de confianza del valor promedio de Y se va ampliando a medida que nos alejamos del centro de gravedad de la nube de punto, donde reside la mayor cantidad de información entre Y y X .

La ordenada en el origen (β_0) y la pendiente (β_1) de la recta se obtienen en R utilizando la función `lm`:

```
ajuste=lm(weight~weeks,data=births)
ajuste
```

```
##
## Call:
## lm(formula = weight ~ weeks, data = births)
##
## Coefficients:
## (Intercept)      weeks
## -7.3120        0.3725
```

Ejercicio 2.1 Contrastar mediante la regresión lineal simple el efecto del consumo de tabaco durante el embarazo sobre el peso del neonato. Comparar con el resultado anteriormente obtenido con el test de Student.

Una información más detallada sobre el ajuste se obtiene con la función `summary`. Este resumen proporciona, indicadores sobre la significatividad de los coeficientes del modelo (standard error y p-valores):

```
summary(ajuste)
```

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

```

## lm(formula = weight ~ weeks, data = births)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -3.2900 -0.7542  0.0851  0.6576  2.6576 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -7.31198   1.26305 -5.789 4.08e-08 ***
## weeks        0.37248   0.03268 11.396 < 2e-16 ***
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1.096 on 148 degrees of freedom
## Multiple R-squared:  0.4674, Adjusted R-squared:  0.4638 
## F-statistic: 129.9 on 1 and 148 DF,  p-value: < 2.2e-16

```

Esta salida incluye también información sobre la calidad del ajuste del modelo (R-squared o *coeficiente de determinación*). Este coeficiente corresponde a la proporción de variabilidad explicada por el modelo y se define como :

$$R^2 = 1 - \frac{RSS}{TSS}$$

donde $TSS = \sum_{i=1}^n (y_i - \bar{y})^2$ es la variabilidad de los datos y $RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ es la variabilidad residual del modelo, es decir una medida de la distancia entre los datos (los y_i) y sus predicciones en la recta de regresión (los $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$).

```

## Cálculo a mano del coeficiente de determinación R^2
y=births$weight
pred.y=ajuste$fitted #predicciones en la muestra
residuos=ajuste$residuals # y- pred.y
TSS= sum((y-mean(y))^2)
RSS= sum(residuos^2)
1-RSS/TSS # R-squared

```

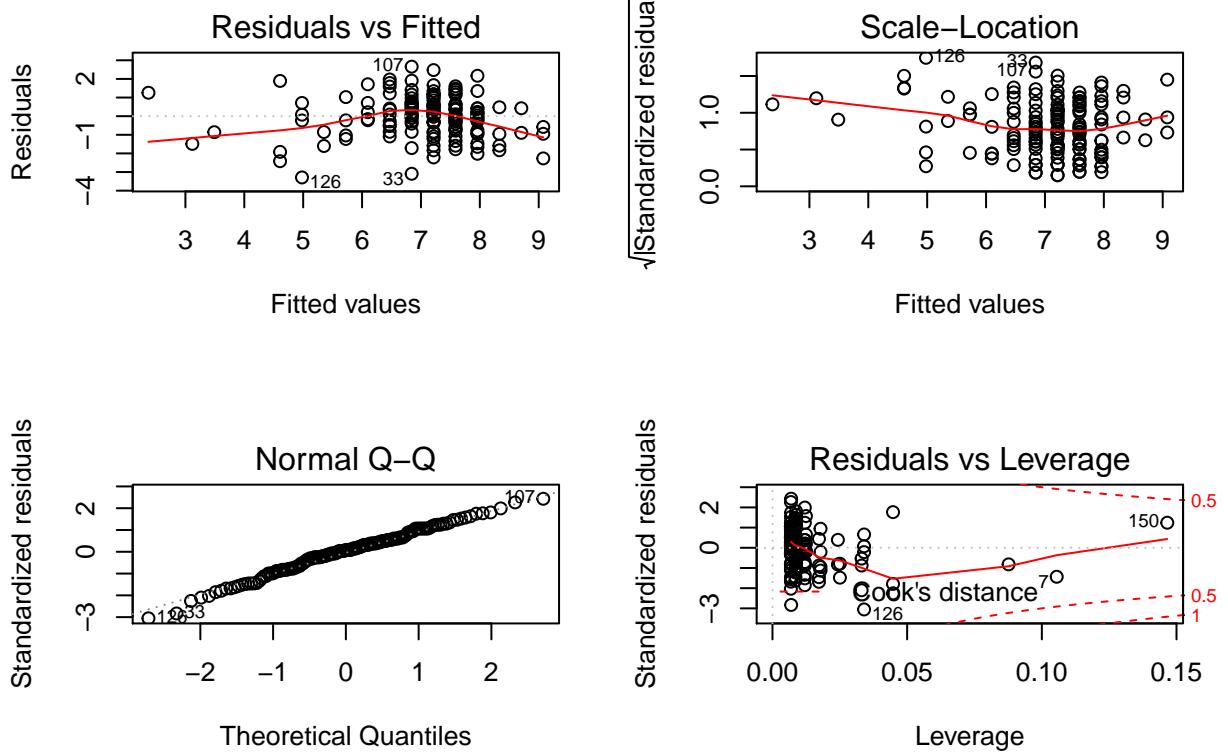
```
## [1] 0.4673912
```

Por otra parte, se puede chequear las asunciones del modelo (residuos con media cero y variaciones constante y aproximadamente normal) mediante la función `plot`:

```

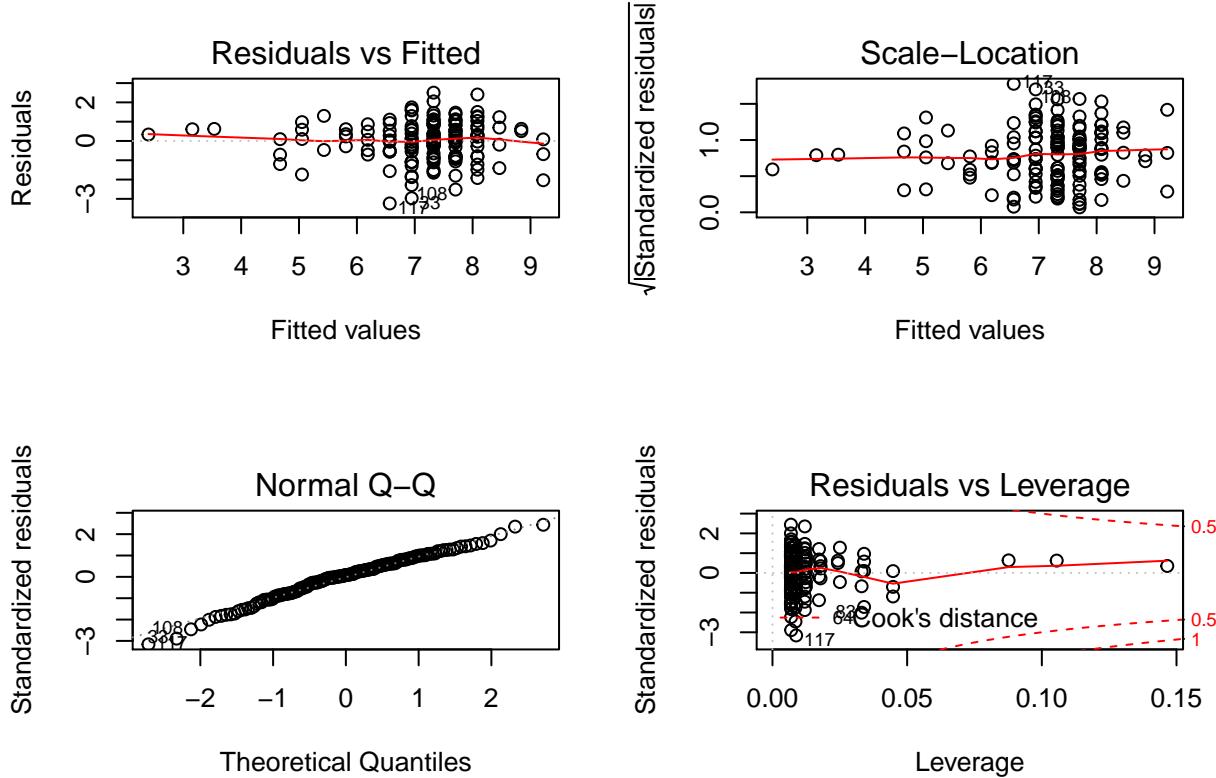
layout(matrix(1:4,2,2))
plot(ajuste)

```



A continuación, estos mismos gráficos en el caso de que se cumplan dichas asunciones:

```
births$weight.sim=simulate(ajuste)$sim_1
layout(matrix(1:4, 2, 2))
plot(lm(weight.sim~weeks, data=births))
```



```
births$weight.sim<-NULL #se quita la simulación de la base
```

Ejercicio 2.2 Construir ejemplos simulados basados en el ejemplo anterior en los cuales no se cumplen cada una de las asunciones sobre los residuos (media cero, homoscedasticidad y normalidad) y observar las consecuencias en los gráficos de validación del modelo.

2.2. Regresión lineal múltiple

El modelo de regresión lineal simple se extiende directamente al caso de varios predictores (X_1, X_2, \dots, X_k) asumiendo

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k + \varepsilon,$$

donde $\varepsilon \sim N(0, \sigma)$.

Los coeficientes del modelo tienen la siguiente interpretación:

- El coeficiente β_0 es el valor promedio de Y cuando $X_1 = X_2 = \dots = X_k = 0$.
- Para cada $j = 1, 2, \dots, k$, el coeficiente β_j corresponde al incremento medio de Y cuando X_j aumenta de una unidad y los demás predictores permanecen constante.
- La desviación típica residual σ corresponde a la variabilidad de Y cuando los predictores permanecen constante.

La implementación de este modelo en R utiliza la misma función `lm`. Así, para predecir el peso de los neonatos de la base de datos `births` utilizando las demás variables de la base, utilizaremos el código:

```
ajuste.mult=lm(weight~., data=births)
summary(ajuste.mult)
```

```

##
## Call:
## lm(formula = weight ~ ., data = births)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.24246 -0.60237 -0.01238  0.55659  2.64613
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.417784  2.211710 -1.545  0.1252
## fAge         0.013317  0.022946  0.580  0.5629
## mAge         0.019595  0.023325  0.840  0.4027
## weeks        0.234228  0.055597  4.213 5.24e-05 ***
## prematurepremie -0.923977  0.449108 -2.057  0.0421 *
## visits        0.035536  0.029403  1.209  0.2295
## gained        0.010025  0.006228  1.610  0.1104
## sexBabymale   -0.062354  0.193431 -0.322  0.7478
## smokesmoker   -0.129076  0.207627 -0.622  0.5355
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9854 on 108 degrees of freedom
## (33 observations deleted due to missingness)
## Multiple R-squared:  0.5382, Adjusted R-squared:  0.504
## F-statistic: 15.73 on 8 and 108 DF,  p-value: 3.633e-15

```

Esta salida del modelo permite en particular averiguar cuales de dichos predictores influyen significativamente sobre el peso de los neonatos.

2.3. Aplicaciones del modelo de regresión múltiple

El modelo de regresión múltiple permite estimar el efecto de cada predictor cuando los demás permanecen constante: **efecto ajustado**. Es además un modelo muy flexible y versátil. A continuación, ilustramos estas propiedades con algunos ejemplos.

Control de la confusión

Ejemplo: Indice de masa corporal (IMC) versus Renta:

```
#####
## Datos de IMC según renta y pais de residencia del individuo
datos=read_csv("data/obesidad.csv")
datos
```

```

## # A tibble: 9,870 x 3
##   renta    imc region
##   <dbl>   <dbl> <chr>
## 1 3.17    24.8 Asia
## 2 7.19    14.3 Asia
## 3 27.2    19.4 Asia
## 4 9.85    23.1 Asia
## 5 22.1    23.6 Europa
## 6 18.5    27.2 Europa
## 7 6.05    16.1 Asia
## 8 22.8    23.6 Europa
## 9 21.9    20.6 Asia

```

```
## 10 12.4 20.8 Asia
## # ... with 9,860 more rows
```

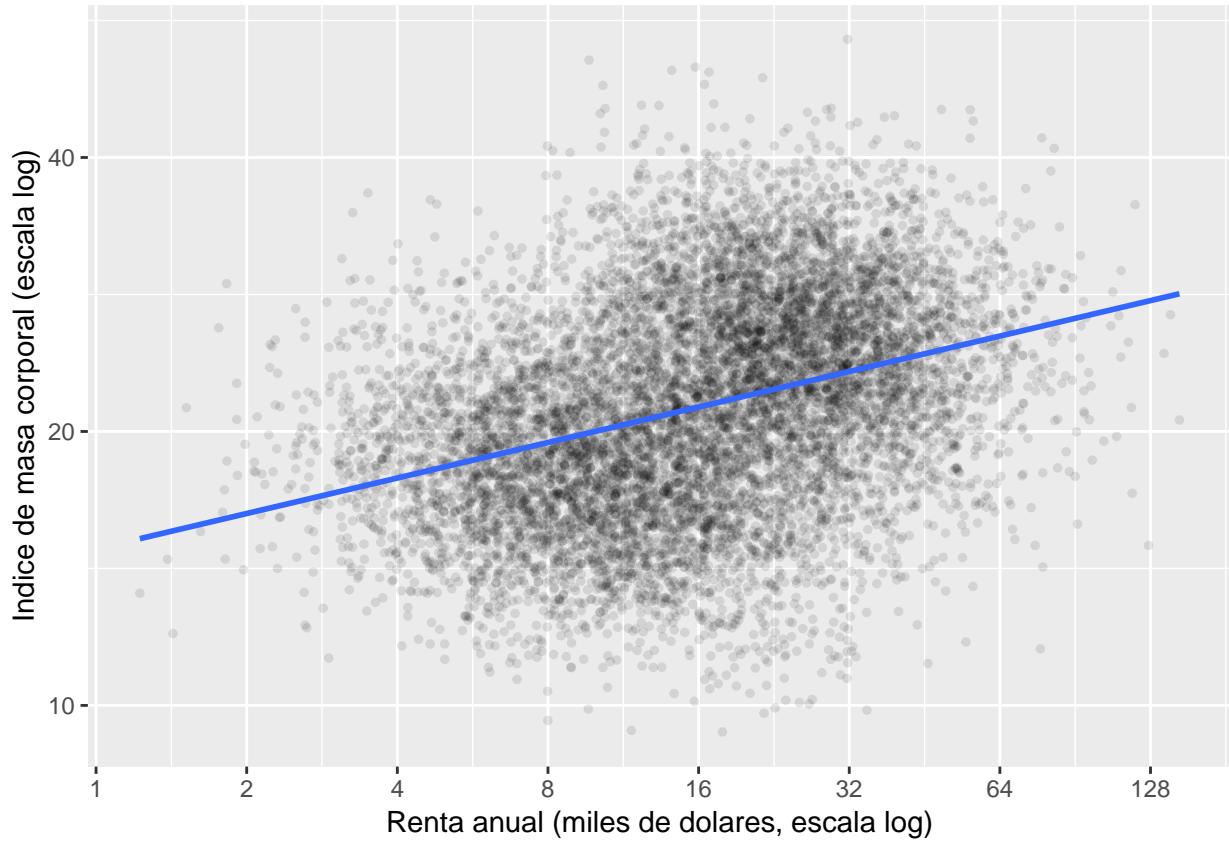
Ajuste con modelo lineal simple: efecto crudo de la renta

$$\log \text{IMC} = \beta_0 + \beta_1 \log \text{Renta} + \varepsilon$$

```
fit=lm(log(imc)~log(renta),data=datos)
summary(fit)
```

```
##
## Call:
## lm(formula = log(imc) ~ log(renta), data = datos)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -0.86700 -0.17188  0.00182  0.17200  0.94312 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.698329  0.010153 265.76 <2e-16 ***
## log(renta)  0.129536  0.003562  36.37 <2e-16 ***  
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.253 on 9868 degrees of freedom
## Multiple R-squared:  0.1182, Adjusted R-squared:  0.1181 
## F-statistic: 1323 on 1 and 9868 DF, p-value: < 2.2e-16

p <- ggplot(datos, aes(x = renta, y = imc))
p <- p + geom_point(alpha=.1,size=1)
p <- p + scale_y_continuous("Indice de masa corporal (escala log)",trans="log2",breaks=10*2^(0:3))
p <- p + scale_x_continuous("Renta anual (miles de dolares, escala log)", trans="log2",breaks=2^(0:7))
p + geom_smooth(method="lm",se = FALSE)
```



IMC versus Renta ajustado por Region

$$\log \text{IMC} = \beta_0 + \beta_1 \log \text{Renta} + \beta_2 \text{Region} + \varepsilon$$

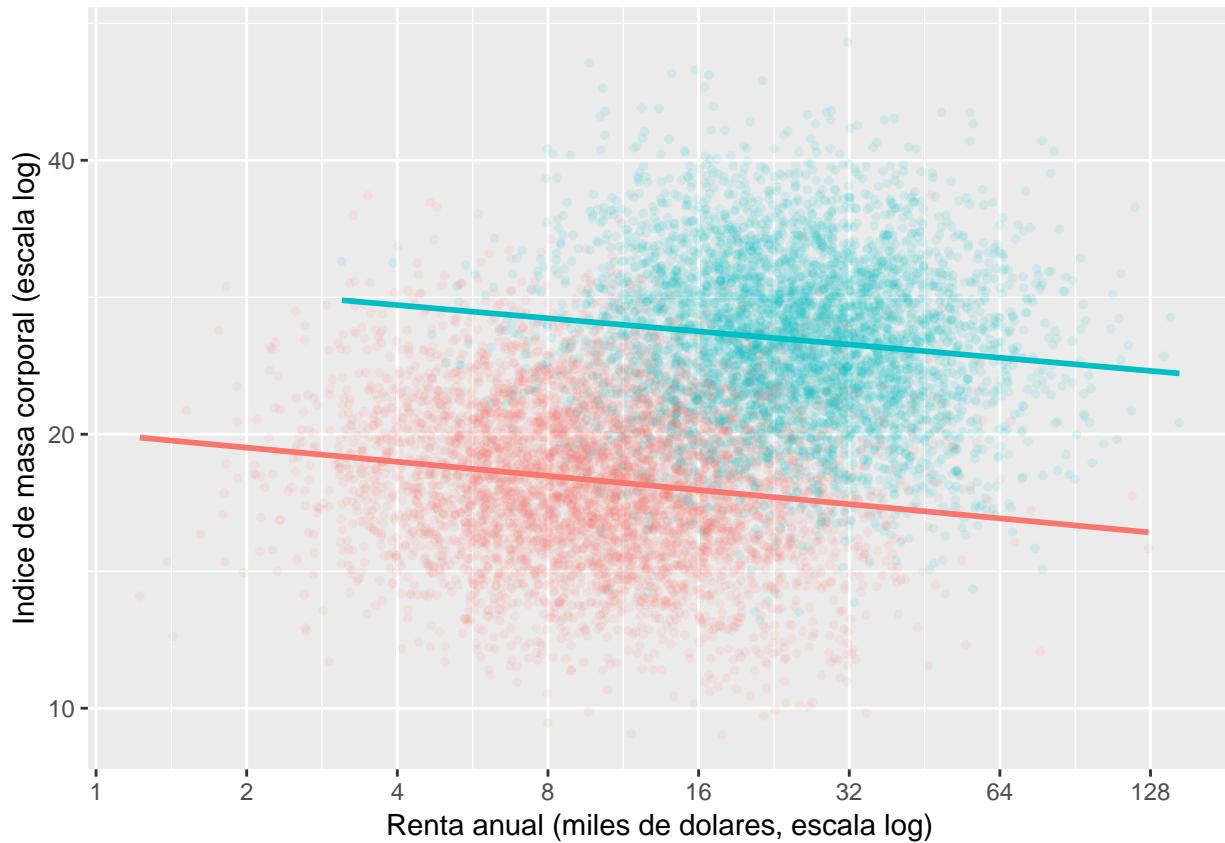
```
###  
fit=lm(log(imc)-log(renta)+region,data=datos)  
summary(fit)

##  
## Call:  
## lm(formula = log(imc) ~ log(renta) + region, data = datos)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -0.71724 -0.13527  0.00032  0.13449  0.79962  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 2.994332  0.008926 335.48 <2e-16 ***  
## log(renta)  -0.050173  0.003671 -13.67 <2e-16 ***  
## regionEuropa 0.401674  0.005249  76.52 <2e-16 ***  
## ---  
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.2004 on 9867 degrees of freedom  
## Multiple R-squared: 0.4466, Adjusted R-squared: 0.4465
```

```

## F-statistic: 3982 on 2 and 9867 DF, p-value: < 2.2e-16
p <- ggplot(datos, aes(x = renta, y = imc, color = region))
p <- p + geom_point(alpha=.1,size=1)
p <- p + scale_y_continuous("Indice de masa corporal (escala log)",trans="log2",breaks=10*2^(0:3))
p <- p + scale_x_continuous("Renta anual (miles de dolares, escala log)", trans="log2",breaks=2^(0:7))
p <- p + geom_smooth(method="lm",se = FALSE)
p + scale_color_discrete("Region",breaks=0:1,labels=c("Europa","Asia"))

```



Modificación del efecto (interacción)

Ejemplo: Peso del neonato versus edad gestacional:

```

##### Ejemplo de Interacción (datos simulados)
set.seed(1234)
n=10000
fumadora=rbinom(n,1,.5) #madre fumadora (1) y no fumadora (0),
gestacion=32+10*runif(n) #tiempo de gestación en semanas
peso=2000 + (gestacion-32)*(fumadora*100 + (1-fumadora)*150)+100*rnorm(n) # peso del neonato (g)
datos <- data_frame(peso=peso,fumadora=fumadora,gestacion=gestacion)
datos

## # A tibble: 10,000 x 3
##       peso fumadora gestacion
##   <dbl>     <int>      <dbl>
## 1 2125.        0       34.0
## 2 2547.        1       36.8
## 3 2688.        1       38.4

```

```

## 4 2973.      1     41.6
## 5 2324.      1     33.8
## 6 2239.      1     34.9
## 7 2368.      0     35.9
## 8 2101.      0     32.8
## 9 2248.      1     34.9
## 10 2480.     1     36.7
## # ... with 9,990 more rows

```

Ajuste con modelo lineal simple:

$$\text{Peso} = \beta_0 + \beta_1 \text{Edad} + \varepsilon$$

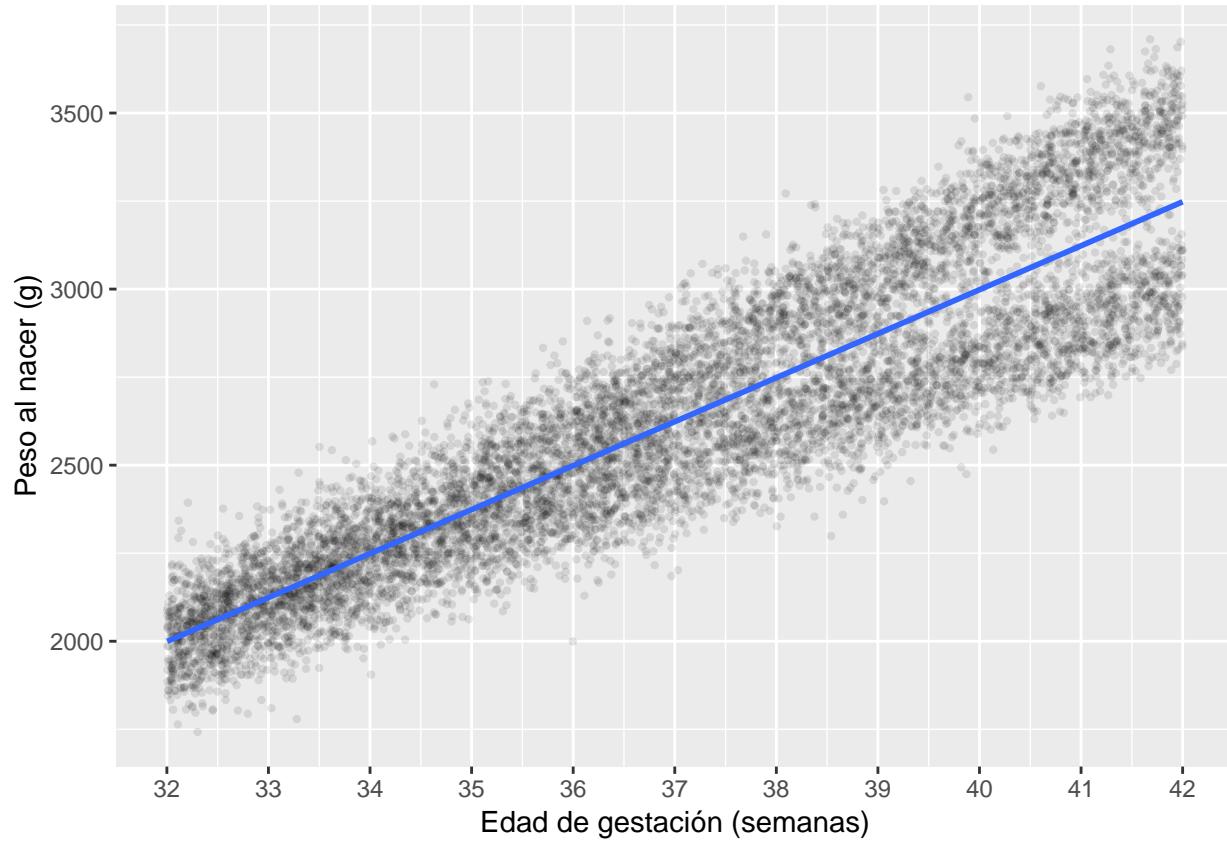
```

##### Ajuste con modelo lineal simple
fit=lm(peso~gestacion,data=datos)
summary(fit)

##
## Call:
## lm(formula = peso ~ gestacion, data = datos)
##
## Residuals:
##       Min     1Q   Median     3Q    Max
## -516.86 -133.51     0.35  129.69  560.58
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1996.7466   22.8552 -87.36 <2e-16 ***
## gestacion     124.8734    0.6151 203.01 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 176.3 on 9998 degrees of freedom
## Multiple R-squared:  0.8048, Adjusted R-squared:  0.8048
## F-statistic: 4.121e+04 on 1 and 9998 DF, p-value: < 2.2e-16

p <- ggplot(datos, aes(x = gestacion, y = peso))
p <- p + geom_point(alpha=.1,size=.8)
p <- p + scale_y_continuous("Peso al nacer (g)")
p <- p + scale_x_continuous("Edad de gestación (semanas)",breaks=32:42)
p + geom_smooth(method="lm",se = FALSE)

```



Modelo con interacción (modificación del efecto de la edad gestacional):

$$\text{Peso} = \beta_0 + \beta_1 \text{Edad} + \beta_2 \text{Edad} \times \text{Fumadora} + \varepsilon$$

```
##### Ajuste con modelo que incluye una interacción entre
fit=lm(peso~gestacion*fumadora,data=datos)
summary(fit)
```

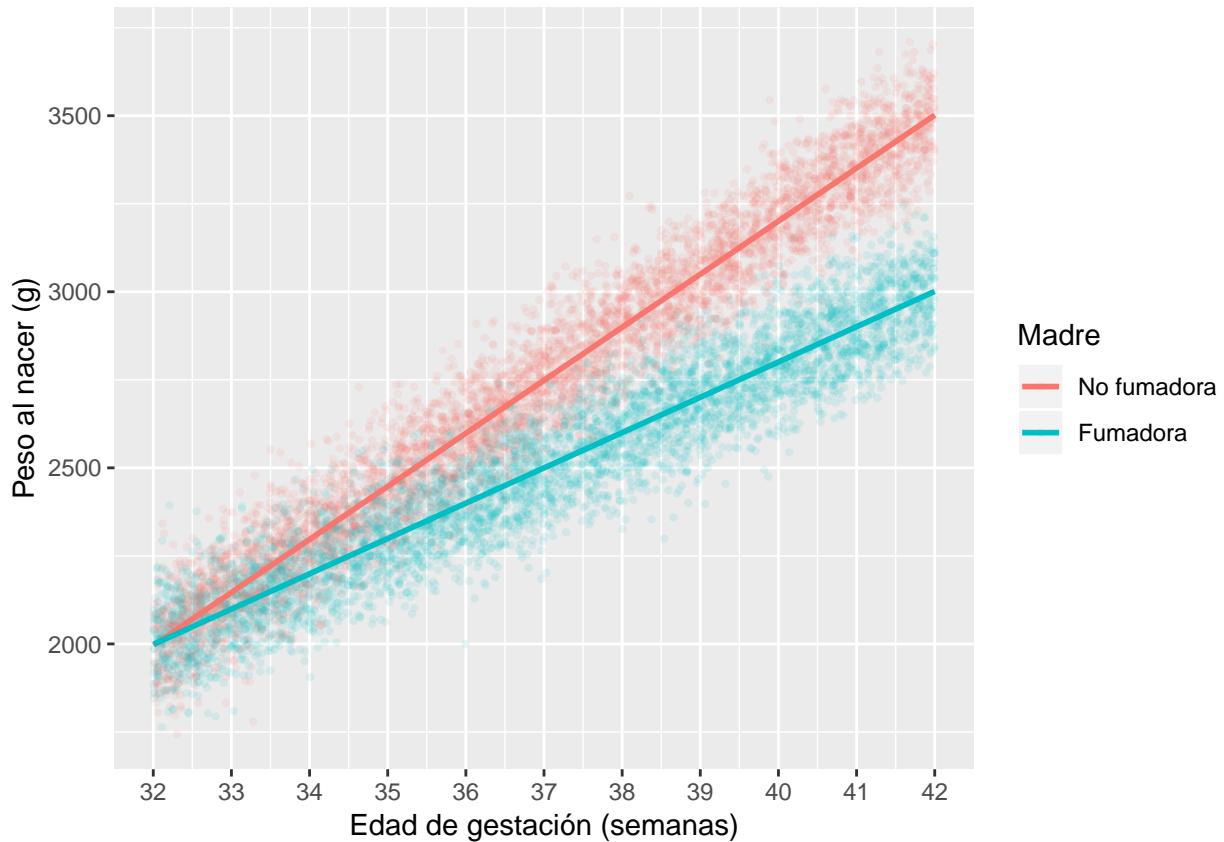
```
##
## Call:
## lm(formula = peso ~ gestacion * fumadora, data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -408.94   -69.76    1.12    68.27  374.03
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2824.1646    18.5470 -152.27 <2e-16 ***
## gestacion     150.6144     0.4994  301.59 <2e-16 ***
## fumadora     1615.1148    26.0846   61.92 <2e-16 ***
## gestacion:fumadora -50.3739     0.7020  -71.75 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 100.6 on 9996 degrees of freedom
```

```

## Multiple R-squared:  0.9364, Adjusted R-squared:  0.9364
## F-statistic: 4.91e+04 on 3 and 9996 DF, p-value: < 2.2e-16

p <- ggplot(datos, aes(x = gestacion, y = peso, color=factor(fumadora)))
p <- p + geom_point(alpha=.1,size=.8)
p <- p + scale_y_continuous("Peso al nacer (g)")
p <- p + scale_x_continuous("Edad de gestación (semanas)",breaks=32:42)
p <- p + geom_smooth(method="lm",se = FALSE)
p + scale_color_discrete("Madre",breaks=0:1,labels=c("No fumadora","Fumadora"))

```



Regresión polinómica

Ejemplo: IMC versus Edad en los jóvenes:

```

##### Ejemplo de relación no lineal (datos simulados)
set.seed(1234)
n=10000
edad=18*round(runif(n),2)
imc=12+5*exp(-(edad-1)^2/4)+10*exp(-(edad-17)^2/200)+rnorm(n)
datos <- data.frame(imc=imc,edad=edad)

```

$$\text{IMC} = \beta_0 + \beta_1 \text{Edad} + \varepsilon$$

```

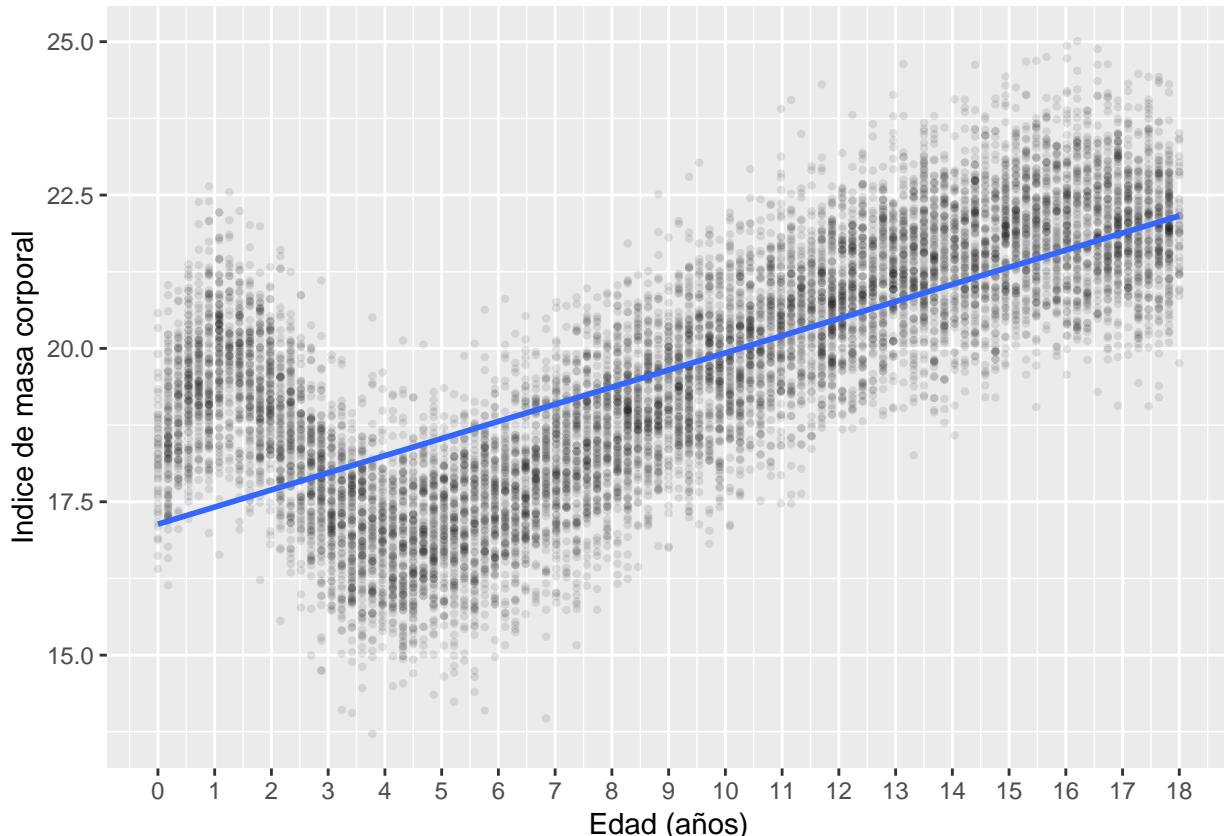
##### Ajuste con regresión lineal simple
fit=lm(imc~edad,data=datos)
summary(fit)

```

```

## 
## Call:
## lm(formula = imc ~ edad, data = datos)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -5.0769 -0.9443 -0.0369  0.9183  5.2546 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 17.135172   0.027856   615.1 <2e-16 ***
## edad         0.279187   0.002683   104.1 <2e-16 ***  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.386 on 9998 degrees of freedom
## Multiple R-squared:  0.52, Adjusted R-squared:  0.5199 
## F-statistic: 1.083e+04 on 1 and 9998 DF, p-value: < 2.2e-16
p <- ggplot(datos, aes(x = edad, y = imc))
p <- p + geom_point(alpha=.1,size=.8)
p <- p+scale_y_continuous("Indice de masa corporal")
p <- p+scale_x_continuous("Edad (años)",breaks=0:18)
p + geom_smooth(method="lm",se = FALSE)

```



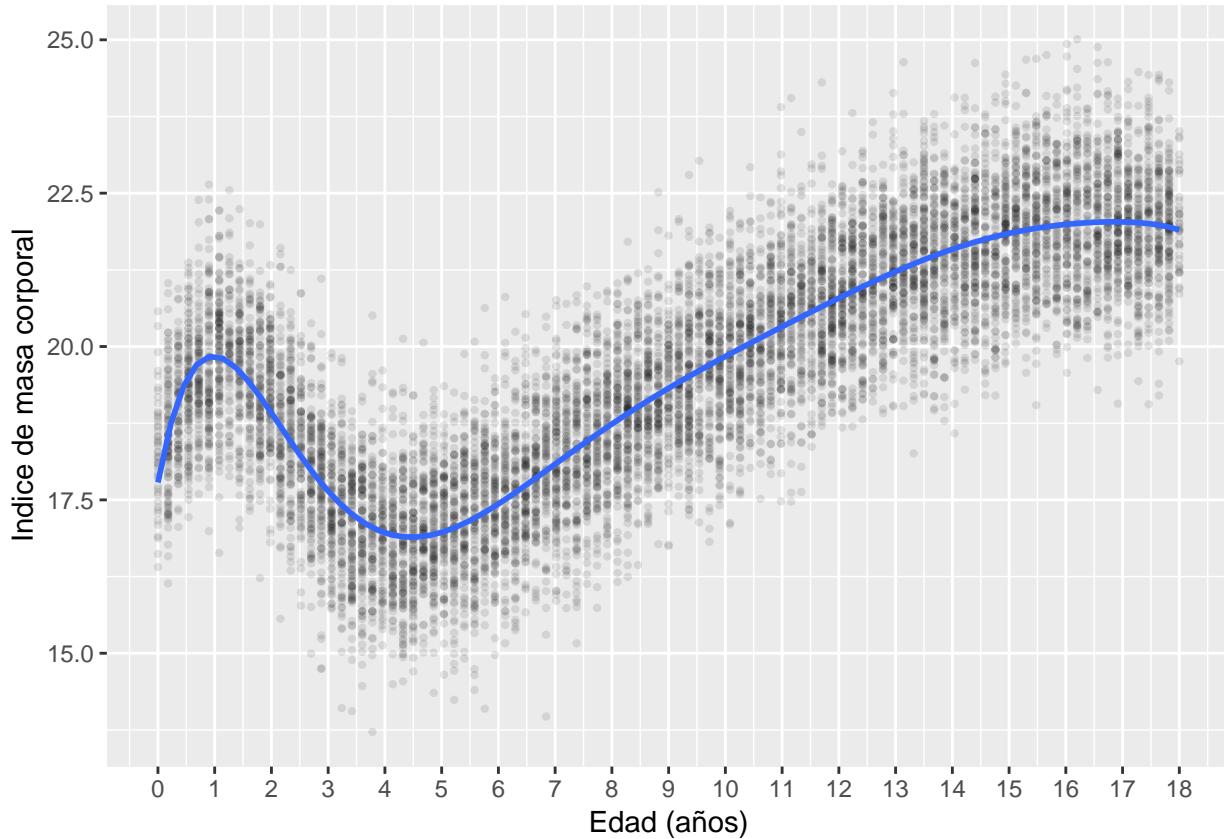
IMC versus potencias de la Edad:

$\text{IMC} = \beta_0 + \beta_1 \text{Edad} + \beta_2 \text{Edad}^2 + \dots + \beta_k \text{Edad}^k + \varepsilon$

```
#####
##### Ajuste con regresión polinómica
fit=lm(imc~poly(edad,degree=8),data=datos)
summary(fit)

##
## Call:
## lm(formula = imc ~ poly(edad, degree = 8), data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -4.0079 -0.6684  0.0012  0.6695  3.6540 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 19.649615  0.009931 1978.666 < 2e-16 ***
## poly(edad, degree = 8)1 144.259003  0.993074 145.265 < 2e-16 ***
## poly(edad, degree = 8)2  55.825404  0.993074  56.215 < 2e-16 ***
## poly(edad, degree = 8)3 -68.365045  0.993074 -68.842 < 2e-16 ***
## poly(edad, degree = 8)4   5.000967  0.993074    5.036 4.84e-07 ***
## poly(edad, degree = 8)5  20.150696  0.993074   20.291 < 2e-16 ***
## poly(edad, degree = 8)6 -27.557849  0.993074  -27.750 < 2e-16 ***
## poly(edad, degree = 8)7  18.513660  0.993074   18.643 < 2e-16 ***
## poly(edad, degree = 8)8 -5.987973  0.993074   -6.030 1.70e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9931 on 9991 degrees of freedom
## Multiple R-squared:  0.7538, Adjusted R-squared:  0.7536 
## F-statistic: 3824 on 8 and 9991 DF,  p-value: < 2.2e-16

p <- ggplot(datos, aes(x = edad, y = imc))
p <- p + geom_point(alpha=.1,size=.8)
p <- p+scale_y_continuous("Indice de masa corporal")
p <- p+scale_x_continuous("Edad (años)",breaks=0:18)
p + geom_smooth(method="lm",se = FALSE,formula = y ~ poly(x,degree=8))
```



```
# p + geom_smooth(method="lm", se = FALSE, formula = y ~ cut(x, 8)) #ajuste con función escalonada (step-f
```

Variables politómicas

Ejemplo: Colesterol (HDL, el “bueno”) versus Tabaquismo:

```
datos = read_csv("data/euramic.csv")
datos = datos %>% filter(statcc==0, !is.na(tipfum))
```

$$\text{IMC} = \beta_0 + \beta_1 \text{FUMPAS} + \beta_2 \text{FUMACT} + \varepsilon$$

```
##### Regresión con variable politomica
fit=lm(hdlcol~factor(tipfum), data=datos)
#fit = lm(hdlcol~ fumpas + fumact, data=datos) #equivalente al modelo anterior
summary(fit)
```

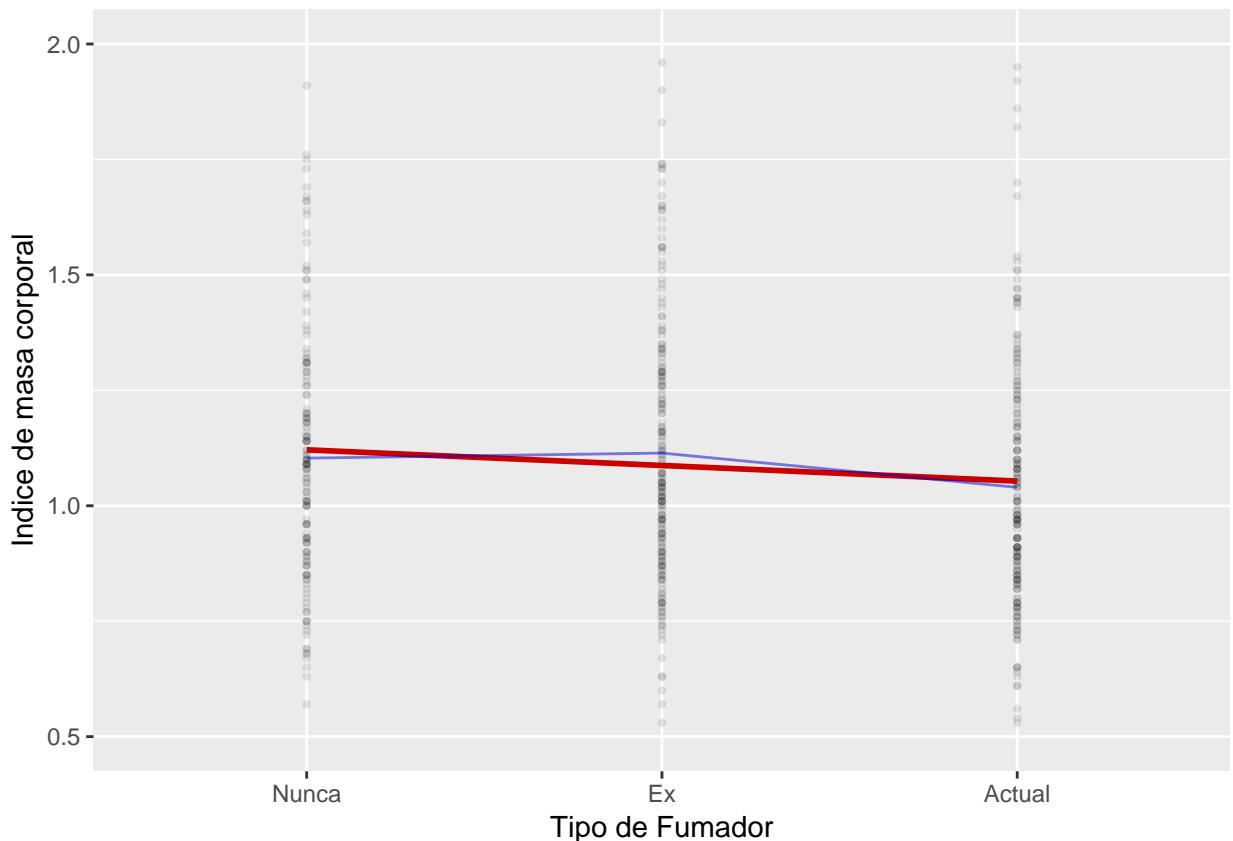
```
##
## Call:
## lm(formula = hdlcol ~ factor(tipfum), data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85526 -0.20471 -0.04526  0.16949  1.08474
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```

## (Intercept) 1.109658 0.024235 45.788 <2e-16 ***
## factor(tipfum)2 0.005598 0.032013 0.175 0.861
## factor(tipfum)3 -0.066596 0.032013 -2.080 0.038 *
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2928 on 535 degrees of freedom
## (161 observations deleted due to missingness)
## Multiple R-squared: 0.01312, Adjusted R-squared: 0.009427
## F-statistic: 3.555 on 2 and 535 DF, p-value: 0.02925

p <- ggplot(datos %>% filter(!is.na(tipfum)), aes(x = tipfum, y = hdlcol))
p <- p + geom_point(alpha=.1,size=.8)
p <- p + scale_y_continuous("Indice de masa corporal",limits=c(.5,2))
p <- p + scale_x_discrete("Tipo de Fumador",breaks=1:3,labels=c("Nunca","Ex","Actual"),limits=1:3)
p <- p + geom_smooth(method="lm",se=FALSE,formula=y~x,col="red3",alpha=.5) #regresión lineal simple
p + stat_summary(geom="line",fun.data="mean_se",col="blue3",alpha=.5) #regresión con variable politómica

```



2.4. Selección de los predictores

La inclusión de un gran número de predictores en el modelo dificulta la interpretación y la estimación de sus efectos. A continuación, se expone brevemente (más detalles en el *Cápitulo 5*) como seleccionar los predictores del modelo.

Condición de identificabilidad

Las condiciones mínimas para poder estimar los efectos de los predictores son:

- El tamaño n de la muestra ha de ser superior a $k + 1$, donde k es el número de predictores incluidos en el modelo.
- Un predictor no puede ser una combinación de los demás predictores (**No colinealidad**).

De manera general, es recomendable para obtener estimaciones estables de los coeficientes del modelo, incluir un número moderado de predictores que además sean poco correlacionados.

Ejercicio 2.3 Construir un ejemplo (a partir de la base de datos `births`) donde existe colinealidad y observar sus consecuencias en la estimación.

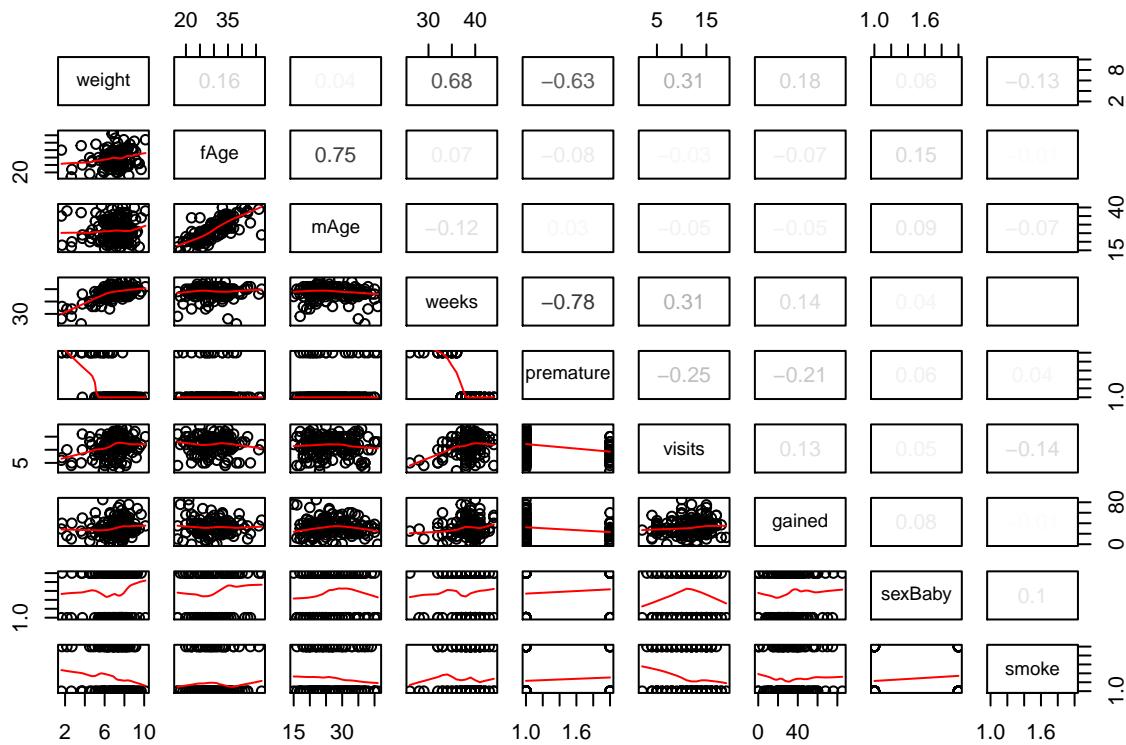
```
##### Matriz de correlación de las variables de la base "births"

births %>% mutate_all(as.numeric) %>% cor(use="pairwise")

##          fAge      mAge      weeks  premature   visits
## fAge 1.0000000 0.75164817 0.065420897 -0.07744508 -0.02830068
## mAge 0.75164817 1.00000000 -0.123705380 0.02602151 -0.04501515
## weeks 0.06542090 -0.12370538 1.000000000 -0.77502387 0.30552083
## premature -0.07744508 0.02602151 -0.775023868 1.00000000 -0.25328821
## visits -0.02830068 -0.04501515 0.305520834 -0.25328821 1.00000000
## gained -0.07299037 -0.04649043 0.140475830 -0.20537648 0.13313663
## weight 0.15945229 0.04230543 0.683660159 -0.63175082 0.31265640
## sexBaby 0.14941413 0.09241450 0.035074828 0.05866363 0.04548386
## smoke -0.00781472 -0.06840597 -0.001721193 0.04075696 -0.13798586
##          gained      weight    sexBaby     smoke
## fAge -0.072990367 0.15945229 0.14941413 -0.007814720
## mAge -0.046490425 0.04230543 0.09241450 -0.068405965
## weeks 0.140475830 0.68366016 0.03507483 -0.001721193
## premature -0.205376481 -0.63175082 0.05866363 0.040756957
## visits 0.133136630 0.31265640 0.04548386 -0.137985862
## gained 1.000000000 0.17500738 0.07857851 -0.008448675
## weight 0.175007378 1.00000000 0.06136735 -0.126521965
## sexBaby 0.078578509 0.06136735 1.00000000 0.104163678
## smoke -0.008448675 -0.12652197 0.10416368 1.000000000

panel.cor<-function(x,y) {
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r=cor(x,y,use="pairwise")
  grises=paste0("grey",round((1-abs(r))*100))
  text(.5,.5,round(r,2),col=grises)}

pairs(weight~.,data=births,lower.panel=panel.smooth,upper.panel=panel.cor)
```



```
#Grafico equivalente con el paquete GGally
#require(GGally)
#ggpairs(births) #más bonito pero lento
```

Criterio de selección del modelo

Para poder seleccionar los predictores, hace falta un criterio que nos permita comparar los distintos modelos. La bondad del ajuste (el R^2 en regresión lineal) no es adecuada porque mejorará siempre a medida que se aumenta el número de predictores. En otras palabras, con este criterio, siempre vamos a seleccionar el modelo que incluye todos los predictores.

La inestabilidad del modelo (de una muestra a otra) crece con el número de predictores. Un criterio más adecuado debería por lo tanto penalizar los modelos que incluyen predictores con poca capacidad predictora (criterio de parsimonia, o *Occam's razor* : “All else being equal, the simplest explanation is the best one”).

Una posible solución es utilizar el criterio de Información de Akaike (AIC) cuya expresión en el modelo de regresión lineal es:

$$AIC = \frac{RSS}{RSS_0} + 2k + (\text{constante})$$

donde RSS_0 es la variabilidad residual en el modelo que incluye todos los predictores.

De acuerdo a este criterio, el “mejor” modelo será el que minimiza esta medida de parsimonia (AIC).

El *Backward Stepwise Selection* es un método de búsqueda ordenada del mejor modelo, que parte del modelo completo y quita sucesivamente los predictores los menos relevantes en la predicción. A continuación ilustramos dicho método mediante la función `step` de R:

```

datos = births %>% subset(select=-fAge) %>% na.omit() #se quita la edad del padre y valores ausentes
# datos = datos %>% mutate(weeks=weeks-39, mAge=mAge-25)
modelo.completo=lm(weight~.,data=datos)
mejor=step(modelo.completo)

## Start: AIC=24.85
## weight ~ mAge + weeks + premature + visits + gained + sexBaby +
##       smoke
##
##          Df Sum of Sq   RSS   AIC
## - sexBaby     1    0.7318 156.86 23.540
## - gained      1    0.7496 156.87 23.557
## <none>          156.12 24.853
## - visits      1    2.6627 158.79 25.339
## - mAge         1    2.9442 159.07 25.599
## - smoke        1    3.4780 159.60 26.092
## - premature    1    5.8286 161.95 28.241
## - weeks        1   29.3517 185.48 48.177
##
## Step: AIC=23.54
## weight ~ mAge + weeks + premature + visits + gained + smoke
##
##          Df Sum of Sq   RSS   AIC
## - gained      1    0.9275 157.78 22.407
## <none>          156.86 23.540
## - visits      1    2.8462 159.70 24.184
## - smoke        1    3.1893 160.05 24.499
## - mAge         1    3.2885 160.15 24.590
## - premature    1    5.3191 162.18 26.443
## - weeks        1   31.1333 187.99 48.156
##
## Step: AIC=22.41
## weight ~ mAge + weeks + premature + visits + smoke
##
##          Df Sum of Sq   RSS   AIC
## <none>          157.78 22.407
## - visits      1    3.1641 160.95 23.326
## - smoke        1    3.1777 160.96 23.338
## - mAge         1    3.1879 160.97 23.347
## - premature    1    6.1751 163.96 26.050
## - weeks        1   30.7028 188.49 46.544

summary(mejor)

##
## Call:
## lm(formula = weight ~ mAge + weeks + premature + visits + smoke,
##      data = datos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -3.5101 -0.6497  0.0102  0.7067  2.6281 
##
## Coefficients:

```

```

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.28564   2.09743 -2.043   0.0429 *
## mAge        0.02439   0.01445  1.688   0.0937 .
## weeks       0.27012   0.05157  5.238 5.79e-07 ***
## prematurepremie -0.93587   0.39840 -2.349   0.0202 *
## visits      0.04348   0.02586  1.682   0.0949 .
## smokesmoker -0.31593   0.18748 -1.685   0.0942 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.058 on 141 degrees of freedom
## Multiple R-squared:  0.5258, Adjusted R-squared:  0.509
## F-statistic: 31.27 on 5 and 141 DF,  p-value: < 2.2e-16

```

Ejercicio 2.4 Utilizando el criterio Akaike y la función `step`, hallar el mejor modelo para la predicción de la esperanza de vida a partir de la información incluida en la base de datos `state.x77`.

2.5. Predicción

Predicción versus Inferencia

En lo anterior, hemos descrito como aproximar la relación $Y = f(X) + \varepsilon$ entre una variable respuesta Y y una lista de predictores $X = \{X_1, X_2, \dots, X_k\}$ mediante una combinación lineal de estas variables:

$$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \dots + \hat{\beta}_k X_k$$

Tipicamente Y es una variable difícil de medir o controlar, mientras que los predictores (o variables explicativas) son variables fáciles de observar y/o sobre las cuales se puede actuar.

Desde el punto de vista de la *Inferencia*, el objetivo es “explicar” las variaciones de Y y averiguar la magnitud del *efecto* de cada una de las componentes de X sobre estas variaciones. El modelo es por lo tanto un soporte a la interpretación y un potencial generador de conocimiento científico (ejemplo: evaluar el efecto del tabaco sobre el desarrollo del feto).

La *Predicción* tiene en general beneficios más a corto plazo y prácticos (ejemplo: predecir el peso del feto a partir de su volumen en la ecografía). En este caso, no estamos tanto interesados en la interpretación del modelo, sino sobre todo en la precisión de sus predicciones en una situación real.

Precisión fuera de la muestra

Cuando el objetivo es predecir, se suele medir la precisión de las predicciones del modelo mediante el error cuadrático medio (MSE, en inglés) que se define como la distancia media entre las observaciones y_i y las predicciones $\hat{f}(x_i) = \hat{y}_i$ del modelo:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}(x_i))^2 = (1 - R^2) \frac{\text{TSS}}{n}$$

Este error está calculado a partir de los datos de la muestra (*training error*). Pero, la vocación de esta muestra es sólo adiestrar el modelo para predecir “situaciones nuevas”. Por lo tanto, estamos más bien interesados en evaluar el MSE en una muestra de datos nuevos (*test error*):

$$\text{MSE}_0 = E(y_0 - \hat{f}(x_0))^2 \simeq \frac{1}{n_0} \sum_{i=1}^{n_0} (y_{0i} - \hat{f}(x_{0i}))^2$$

donde los $(y_{0i}, x_{0i})_i$ denotan un conjunto de datos nuevos.

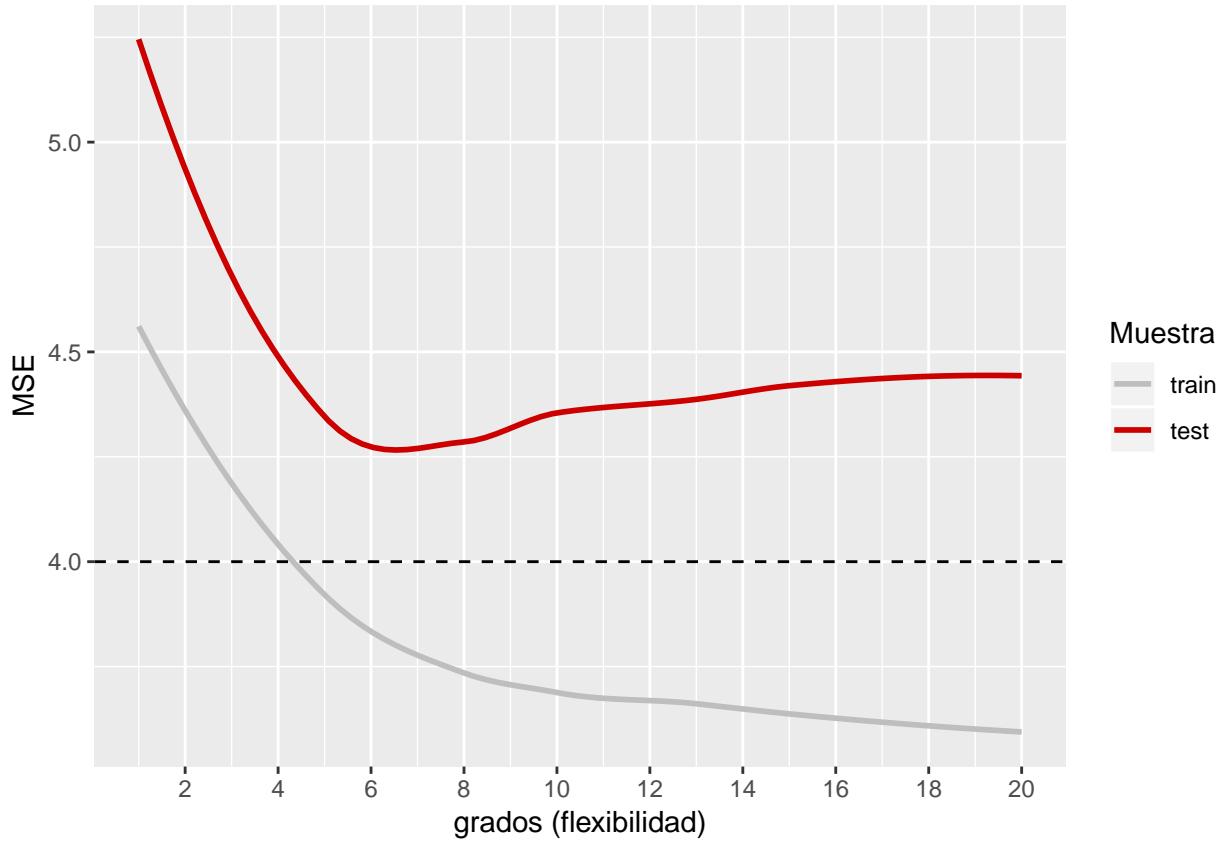
En R la función genérica para la predicción es `predict`. A continuación se evalúa el *training error* y el *test error* utilizando los datos simulados sobre IMC y edad entre los jóvenes para modelos de regresión polinómica de distintos grados.

```
##### Simulación de los datos
set.seed(1234)
n=600
edad=18*round(runif(n),2)
imc=12+5*exp(-(edad-1)^2/4)+10*exp(-(edad-17)^2/200)+2*rnorm(n)
datos <- data.frame(imc=imc,edad=edad)

index = sample(c(TRUE,FALSE),nrow(datos),replace=TRUE,prob=c(2/3,1/3))
Train = datos[index, ]
Test = datos[!index, ]

##### Calculo de MSE y MSEO para distintos grados del polinomio (k)

res<-function(k){
  fit=lm(imc~poly(edad,k),data=Train)
  pred=predict(fit) #predicción en la muestra train (ajuste)
  pred0=predict(fit,newdata=Test) #predicción en test
  MSE= mean((Train$imc-pred)^2) #equivalente a mean(fit$residuals^2)
  MSEO=mean((Test$imc-pred0)^2)
  c(k=k,train=MSE,test=MSEO)
}
temp=Vectorize(res)(k=1:20)
Res=as_data_frame(t(temp)) %>% gather(muestra,MSE,-k)
ggplot(Res,aes(x=k,y=MSE,color=muestra))+geom_smooth(se=FALSE) +
  scale_x_continuous("grados (flexibilidad)",breaks=seq(2,20,2)) +
  scale_color_manual("Muestra",values=c("grey","red3"),limits=c("train","test")) +
  geom_abline(intercept=2^2,slope=0,linetype=2) #var(eps)
```



Ejercicio 2.5 De manera similar, evaluar el *test error* en la predicción del peso del neonato mediante la edad de gestación utilizando la base de datos `babies` como muestra de entrenamiento y la base `births` como muestra test (¡cuidado con las unidades!).

Balance entre sesgo y varianza

Acabamos de ver que para adaptarse a una nueva realidad, hace falta encontrar un balance entre flexibilidad y estabilidad del modelo (*Bias- Variance Trade-Off*).

Formalmente, el *test error* admite la siguiente descomposición:

$$\text{MSE}_0 = \text{var}(\hat{f}(x_0)) + \text{Bias}(\hat{f}(x_0))^2 + \text{var}(\varepsilon).$$

- En primer lugar, esta descomposición nos dice que como mínimo MSE_0 será mayor a la variabilidad **no explicada** por X : $\text{var}(\varepsilon)$. Es por lo tanto un error irreductible.
- Lo mejor que podemos hacer es procurar minimizar de manera simultanea tanto el sesgo como la variabilidad de las predicciones (entre muestras de adiestramiento).

Desgraciadamente estas dos cantidades (sesgo y varianza) compiten!! A mayor flexibilidad (o complejidad) del modelo, menor sesgo pero mayor inestabilidad (de una muestra de adiestramiento a otra).

3. Clasificación

3.1. Planteamiento del problema

En el modelo de regresión lineal, se asume que la respuesta Y es cuantitativa. Pero a menudo, Y es una variable categórica (o cualitativa). Por ejemplo, Y puede ser el resultado de un test de embarazo, o de un

control de dopaje, la clasificación de un paciente en urgencia, una cifra escrita a mano que hemos de reconocer, o determinar a partir de una foto si una seta es comestible o toxica!

Predecir Y en este contexto a partir de la observación de X , se llama **clasificación**, porque pretendemos predecir a qué clase/categoría pertenece esta observación.

Cómo en el caso de la predicción en regresión lineal, disponemos de una muestra de adiestramiento $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ a partir de la cual queremos construir un clasificador. Pero, queremos que nuestro clasificador se porte bien no sólo en esta muestra sino también en una nueva muestra-test.

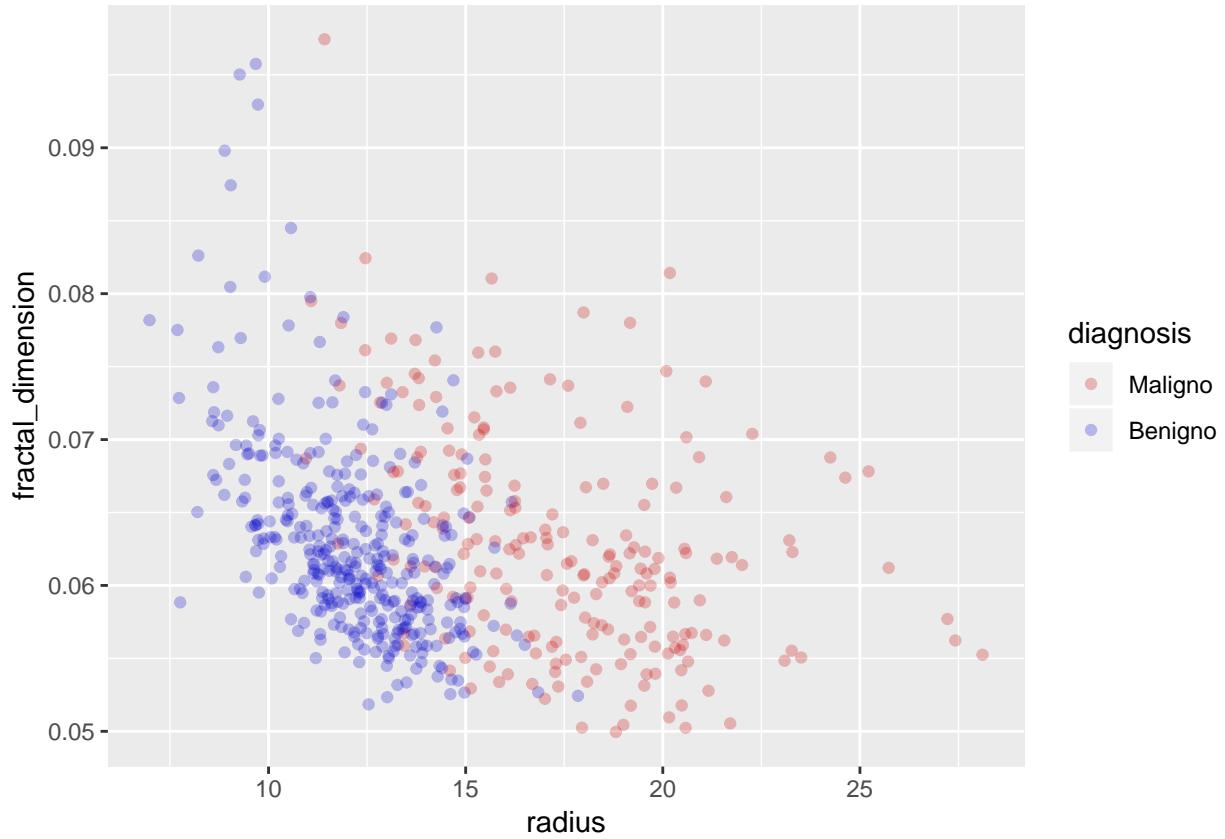
A continuación veremos algunas herramientas de construcción de un clasificador que son la regresión logística, el análisis discriminante y el método de k-vecinos. Para ilustrar estos métodos, utilizaremos la base de datos `mama`, extraída de un estudio de los noventa en Wisconsin sobre diagnóstico del cáncer de mama por imagen (más detalles aquí).

El cáncer de mama es el más común en mujeres y ocurre como resultado del crecimiento anormal de células en el tejido del seno, comúnmente conocido como un tumor. Un tumor no significa cáncer: los tumores pueden ser benignos (B) o malignos (M). El objetivo es aquí determinar si el tumor es maligno a partir de dos características de las células del tejido de la mama.

```
## fuente: http://mlr.cs.umass.edu/ml/datasets/Breast+Cancer+Wisconsin+(Prognostic)
mama=read_csv("data/mama.csv")
glimpse(mama)

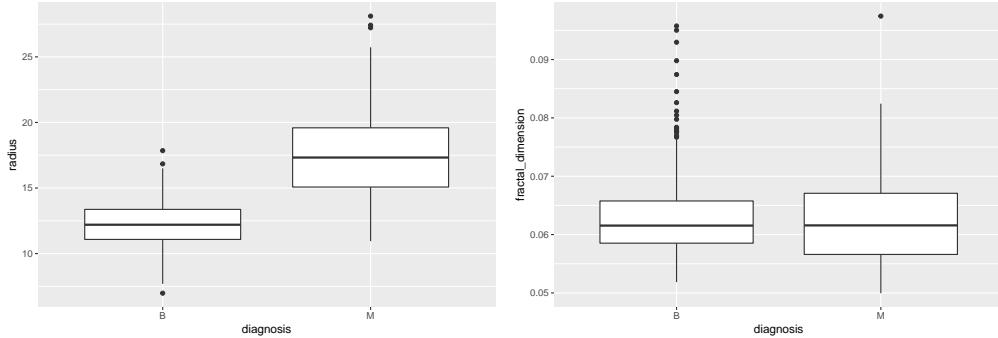
## Observations: 569
## Variables: 10
## $ diagnosis      <chr> "M", "M", "M", "M", "M", "M", "M", "M", ...
## $ radius         <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.4...
## $ texture        <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19...
## $ perimeter      <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57...
## $ smoothness     <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, ...
## $ compactness    <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, ...
## $ concavity      <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, ...
## $ `concave points` <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, ...
## $ symmetry       <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.20...
## $ fractal_dimension <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, ...

ggplot(mama,aes(x=radius,y=fractal_dimension,color=diagnosis)) + geom_point(alpha=.25) +
  scale_color_manual(values=c("red3","blue3"),limits=c("M","B"),labels=c("Maligno","Benigno"))
```



Donde podemos apreciar que la dimensión fractal por si sola caracteriza poco el tumor, mientras que la variable radius separa mejor los tumores malignos de los benignos.

```
qplot(diagnosis, radius, data=mama, geom="boxplot")
qplot(diagnosis, fractal_dimension, data=mama, geom="boxplot")
```



3.2. Un simple Clasificador

Un simple clasificador consistiría en decidir que el tumor es maligno si una de las variables x supera un cierto umbral u :

$$\hat{y}(x) = \begin{cases} \text{Maligno} & x > u \\ \text{Benigno} & x \leq u \end{cases}$$

A la vista de los gráficos anteriores, consideramos que podemos utilizar la variable `radius` para construir un

		Actual	
		False (0)	True (1)
Predicted	False (0)	True Negative (TN)	False Negative (FN)
	True (1)	False Positive (FP)	True Positive (TP)

Figura 2:

clasificador razonable. Para evaluar esta regla de clasificación, dividimos la base de datos en una muestra de adiestramiento (Train) y una muestra de validación (Test):

```
set.seed(4321)
index = sample(c(TRUE,FALSE),nrow(mama),replace=TRUE,prob=c(.5,.5))
Train = mama[index, ]
Test = mama[!index, ]
Train %>% group_by(diagnosis) %>% summarize(mean(radius))

## # A tibble: 2 x 2
##   diagnosis `mean(radius)`
##   <chr>          <dbl>
## 1 B              12.1
## 2 M              17.4
```

El calculo de la media del `radius` en las dos clases de tumores en la muestra Train, no sugiere un umbral de 15.

$$\hat{y}(\text{radius}) = \begin{cases} \text{Maligno} & \text{radius} > 15 \\ \text{Benigno} & \text{radius} \leq 15 \end{cases}$$

Así, predecimos que un tumor es maligno si el `radius` de sus células es mayor de 15, y “benigno” en caso contrario.

```
Test <- Test %>% mutate( pred = ifelse(radius>15,"M","B") )
```

3.3. Medida de error en Clasificación

En clasificación, hay varias métricas para evaluar el desempeño de un clasificador. Una cosa sencilla que se puede hacer, es organizar predicciones y valores reales en una tabla cruzada:

```
tabla=table(prediccion= Test$pred, real = Test$diagnosis)
tabla
```

```
##           real
## prediccion  B    M
##           B 168  21
##           M   7   77
```

En aprendizaje supervisado, esta tabla suele llamarse **Matriz de confusión**.

La función `confusionMatrix()` del paquete `caret` puede ser utilizada para obtener información adicional que damos a continuación para la muestra Test. Ojo que especificamos cual categoría consideramos como “positive”.

```

require(caret)
confusionMatrix(tabla, positive = "M")

## Confusion Matrix and Statistics
##
##           real
## predicción   B   M
##           B 168  21
##           M   7  77
##
##           Accuracy : 0.8974
##                 95% CI : (0.8552, 0.9308)
##     No Information Rate : 0.641
##     P-Value [Acc > NIR] : < 2e-16
##
##           Kappa : 0.7699
## McNemar's Test P-Value : 0.01402
##
##           Sensitivity : 0.7857
##           Specificity : 0.9600
##     Pos Pred Value : 0.9167
##     Neg Pred Value : 0.8889
##           Prevalence : 0.3590
##           Detection Rate : 0.2821
## Detection Prevalence : 0.3077
##     Balanced Accuracy : 0.8729
##
##     'Positive' Class : M
##

```

La medida de error más común en clasificación es el **classification error rate** que simplemente calcula la proporción de fallos en la clasificación.

$$\text{err}(\hat{y}) = \frac{1}{n} \sum_{i=1}^n (y_i \neq \hat{y}_i)$$

Es también habitual considerar la *exactitud* (*accuracy*) del clasificador que no es más que 1 menos el error. En ciertos contextos, es mejor centrarse en los Falsos positivos o Falsos negativos generados por el clasificador, en este caso, nos interesa medidas como la *especificidad* y la *sensibilidad*.*.

$$\text{Sensibilidad} = \text{aciertos entre los positivos} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Especificidad} = \text{aciertos entre los negativos} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Un idea intuitiva, nos lleva a pensar que un clasificador con una tasa de aciertos superior a 0.5 es razonable. Nada más falso, porqué aquí un clasificador que decide sistemáticamente que el tumor es benigno acertaría en el 64% (aquí la prevalencia de casos positivos es de unos 36% en la muestra Test).

A continuación veremos técnicas que permiten la elaboración sistemática de buenos clasificadores.

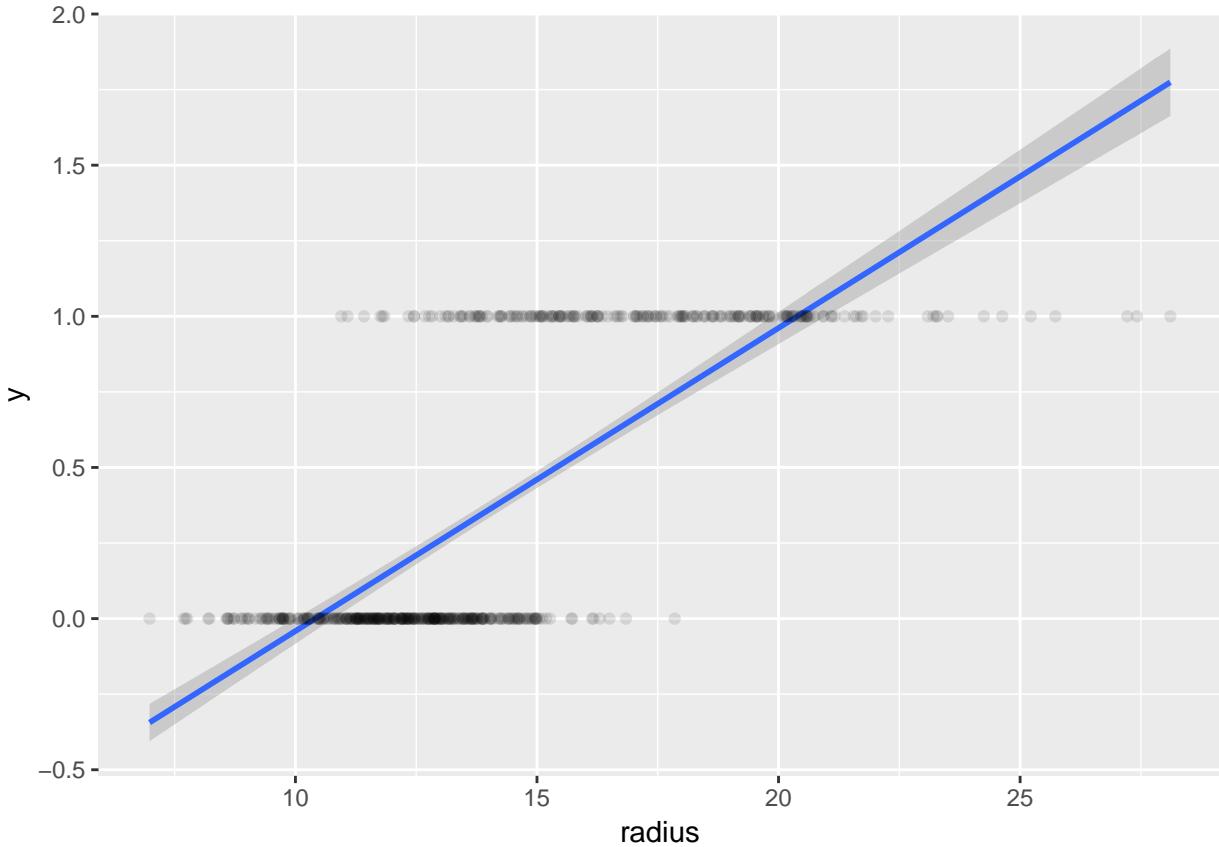
3.4. Regresión logística

El modelo de regresión logística es una extensión del modelo de regresión lineal para variables dicotómicas. Pero antes de presentar esta extensión, porque no primero probar la regresión lineal a ver lo que pasa. Puesto, que este modelo espera una respuesta numérica, codificamos el diagnóstico de la siguiente manera:

$$Y = \begin{cases} 1 & \text{si diagnosis} = 'M' \\ 0 & \text{si diagnosis} = 'B' \end{cases}.$$

Todo parece correcto, pero al ajustar el modelo para estimar la probabilidad $P(Y = 1|X = x)$ obtenemos...

```
mama <- mama %>% mutate(y=ifelse(diagnosis=="M", 1, 0))
ggplot(mama,aes(y=y,x=radius))+geom_smooth(method="lm") +geom_point(alpha=.1)
```



Como se puede apreciar en el gráfico, las predicciones de Y pueden salir del intervalo $[0, 1]$.

Un mejor opción para estimar la probabilidad

$$p(x) = P(Y = 1 | X = x)$$

consiste en suponer la siguiente relación

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p.$$

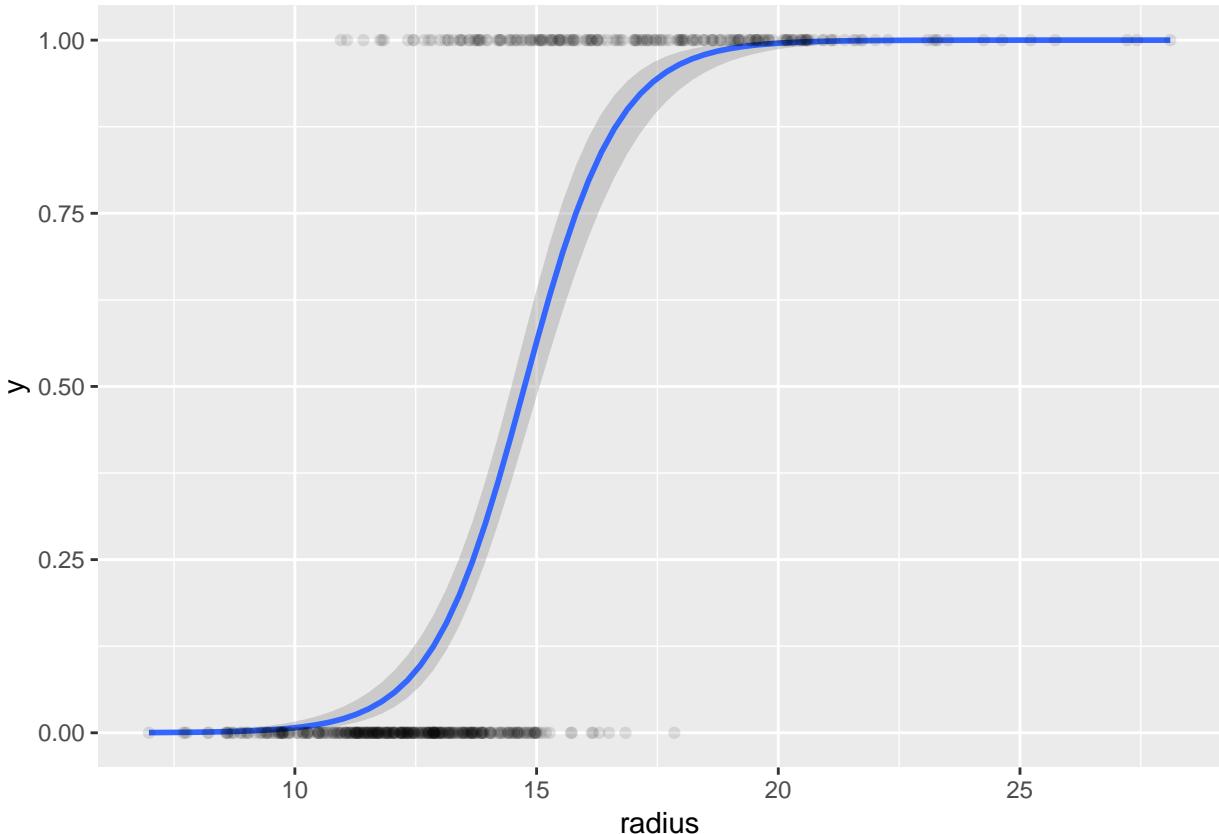
Si despejamos $p(x)$ obtenemos que

. El modelo asume que la siguiente relación entre la probabilidad de $Y = 1$ y el predictor X :

$$p(x) = \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)}{1 + \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)}$$

Puesto que esta cantidad pertenece a $[0, 1]$ para cualquier valor de x , las predicciones de Y estarán bien definidas:

```
ggplot(mama,aes(y=y,x=radius))+
  geom_smooth(method="glm",method.args = list(family = "binomial"))+geom_point(alpha=.1)
```



Ajuste del modelo

Ajustar este modelo es muy similar a ajustar una regresión lineal. En lugar de `lm` usamos `glm`. La única otra diferencia es el uso de `family = "binomial"` que indica que tenemos una respuesta categórica de dos clases:

```
Train = mama[index, ]
Test = mama[!index, ]
fit.glm=glm(y~radius+texture,data=Train,family="binomial")
summary(fit.glm)

##
## Call:
## glm(formula = y ~ radius + texture, family = "binomial", data = Train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max 
## -2.1981   -0.4108   -0.1456    0.1248    2.7338
```

```

## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -18.87230   2.33383 -8.086 6.14e-16 ***
## radius       1.02567   0.14022  7.315 2.58e-13 ***
## texture      0.20155   0.05058  3.985 6.76e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 394.58 on 295 degrees of freedom
## Residual deviance: 157.72 on 293 degrees of freedom
## AIC: 163.72
## 
## Number of Fisher Scoring iterations: 7

```

Los parámetros del modelo admiten una interpretación similar al del modelo de regresión lineal. Así, el coeficiente de `radius` es significativamente positivo e indica por lo tanto que valores crecientes de esta variable aumentan la probabilidad de tumor maligno, como observamos en el gráfico anterior.

Predicción

La estimación de los parámetros del modelo permite predecir una probabilidad de clasificación para cada observación.

```
##### Cálculo de las probabilidades de clasificación
proba=predict(fit.glm,Test,type="response") #predicción de la probabilidad de tumor maligno
```

Sobre la base de estas probabilidades podemos construir un clasificador de la siguiente manera:

$$\hat{y}(x) = \begin{cases} 1 & \hat{p}(x) > 0,5 \\ 0 & \hat{p}(x) \leq 0,5 \end{cases}$$

```
Test$pred=ifelse(proba>.5,"M","B")
tabla=table(prediccion=Test$pred,real=Test$diagnosis) #matriz de confusión
confusionMatrix(tabla)
```

```

## Confusion Matrix and Statistics
##
##           real
## prediccion   B    M
##           B 163  13
##           M  12  85
##
##           Accuracy : 0.9084
##                 95% CI : (0.8678, 0.9399)
##     No Information Rate : 0.641
##     P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.8006
## McNemar's Test P-Value : 1
##
##           Sensitivity : 0.9314
## Specificity  : 0.8673

```

```

##           Pos Pred Value : 0.9261
##           Neg Pred Value : 0.8763
##           Prevalence : 0.6410
##           Detection Rate : 0.5971
##   Detection Prevalence : 0.6447
##           Balanced Accuracy : 0.8994
##
##           'Positive' Class : B
##
#####
##### De manera más directa con ayuda del paquete `caret`
mama$diagnosis=factor(mama$diagnosis) #caret require que la respuesta sea un factor
Train = mama[index, ]
Test = mama[!index, ]
train.glm <- train(diagnosis ~ radius+texture, data = Train, method = "glm", family="binomial")
predicciones=predict(train.glm,Test)
confusionMatrix(predicciones, Test$diagnosis , positive="M")

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    B     M
##           B 163   13
##           M   12   85
##
##           Accuracy : 0.9084
##           95% CI : (0.8678, 0.9399)
##           No Information Rate : 0.641
##           P-Value [Acc > NIR] : <2e-16
##
##           Kappa : 0.8006
##   Mcnemar's Test P-Value : 1
##
##           Sensitivity : 0.8673
##           Specificity : 0.9314
##           Pos Pred Value : 0.8763
##           Neg Pred Value : 0.9261
##           Prevalence : 0.3590
##           Detection Rate : 0.3114
##   Detection Prevalence : 0.3553
##           Balanced Accuracy : 0.8994
##
##           'Positive' Class : M
##

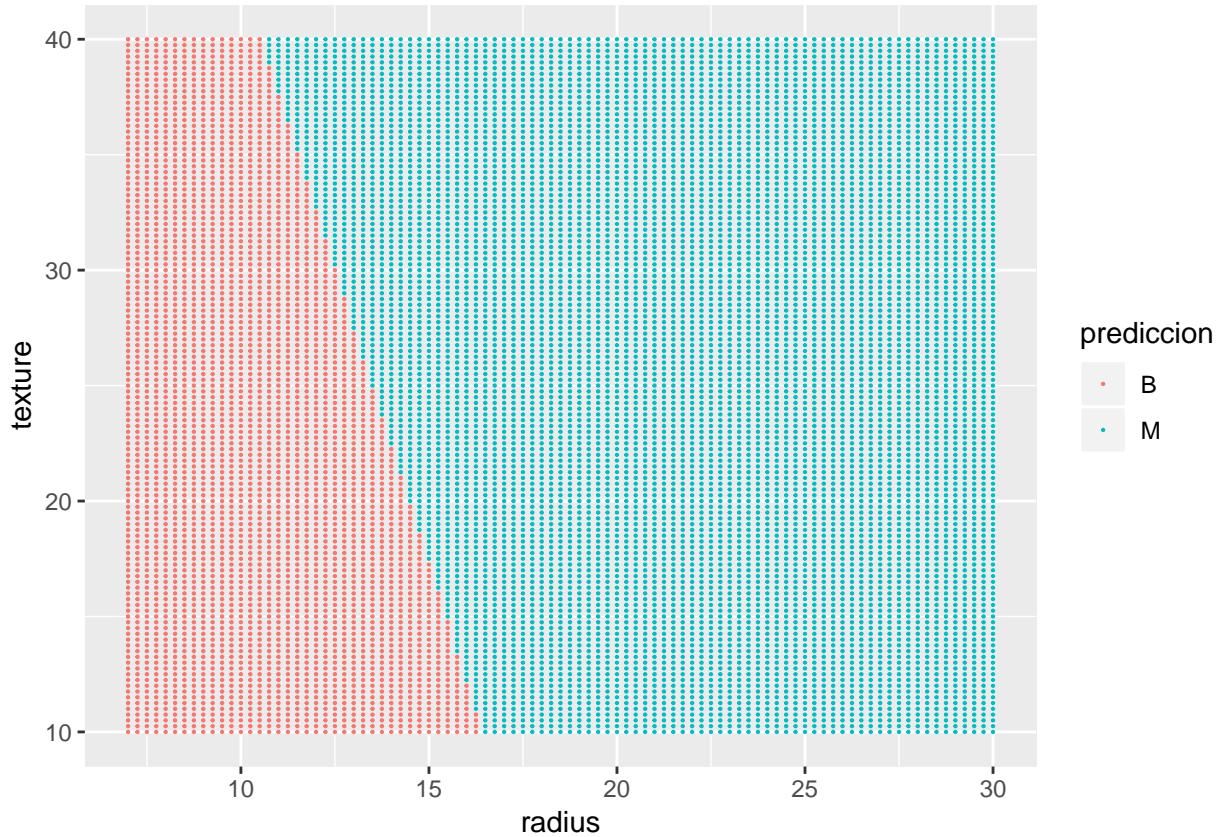
```

Utilizando un grid de los valores de X podemos visualizar la frontera que define el clasificador en el espacio de los predictores:

```

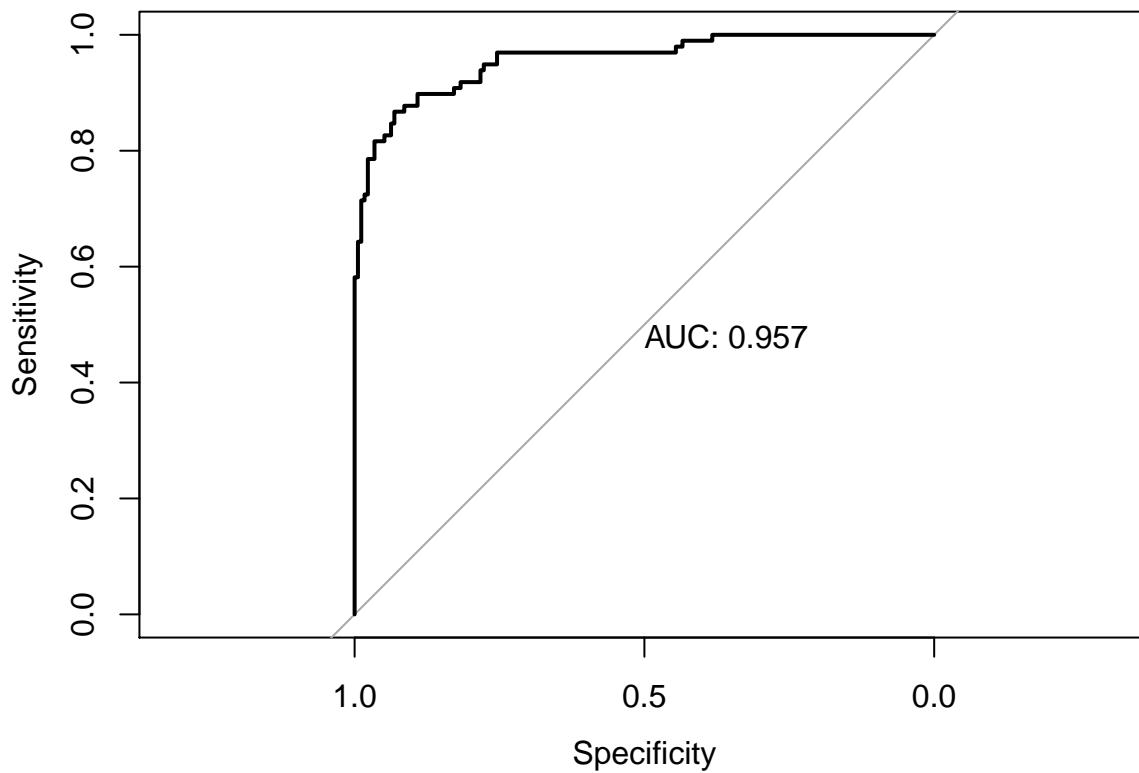
new=expand.grid(radius=seq(7,30,.25),texture=seq(10,40,.25))
new$prediccion=predict(train.glm,new)
ggplot(new,aes(x=radius,y=texture,color=prediccion))+geom_point(size=.1)

```



Acabamos de construir un clasificador, en el cual votamos por “maligno” si la probabilidad $\hat{p}(x)$ supera el umbral 0.5. Sin embargo, podría no ser siempre la mejor elección. Un valor bajo de este umbral aumentaría la *sensibilidad* de la decisión pero reduciría el valor de la *especificidad*. Para poder visualizar en un sólo gráfico, las propiedades del clasificador para distintos valores del umbral, utilizamos el curvas ROC:

```
library(pROC)
test_roc = roc(Test$diagnosis ~ proba, plot = TRUE, print.auc = TRUE)
```



```
head(as.data.frame(test_roc[2:4]))
```

```
##   sensitivities specificities thresholds
## 1           1     0.000000000      -Inf
## 2           1     0.005714286  0.0008649924
## 3           1     0.011428571  0.0011630575
## 4           1     0.017142857  0.0011867611
## 5           1     0.022857143  0.0012453190
## 6           1     0.028571429  0.0013730798
```

Un buen modelo tendrá un alto AUC (medida del área debajo de la curva) y a menudo el mejor compromiso corresponderá a un umbral igual a 0.5.

Ejercicio 3.1 Elaborar un clasificador de tumor utilizando las demás variables que caracterizan las células tumorales de la mama y evaluar su error de predicción en la muestra test.

Ejercicio 3.2 A partir de la base de datos `Smarket` sobre las variaciones del índice bursátil Standard & Poor's de wall street, elaborar un clasificador de alza/baja de este índice sobre la base de su comportamiento en los días anteriores. Escoger el año 2005 como muestra de validación del clasificador.

El modelo de regresión logística se puede en principio extender al caso de más de dos clases. Sin embargo el análisis lineal discriminante ofrece un marco mucho más sencillo para esta extensión.

3.5. Análisis Lineal Discriminante

Supongamos que queremos clasificar una observación en C posibles categorías ($C \geq 2$). En el enfoque del Análisis Lineal Discriminante, se obtiene las probabilidades de clasificación asumiendo que la distribución de

X en cada clase k ($c = 1, 2, \dots, C$) es una normal con una media distinta μ_c (\dots y una misma varianza).

De ello se deduce (utilizando el teorema de Bayes) que

$$P(Y = c|X = x) = \frac{\pi_c f_c(X)}{\sum_{c=1}^C \pi_c f_c(X)}$$

donde las probabilidades π_c son *a priori* sobre cada clase y f_c es la densidad de una normal con media μ_c y una varianza fija.

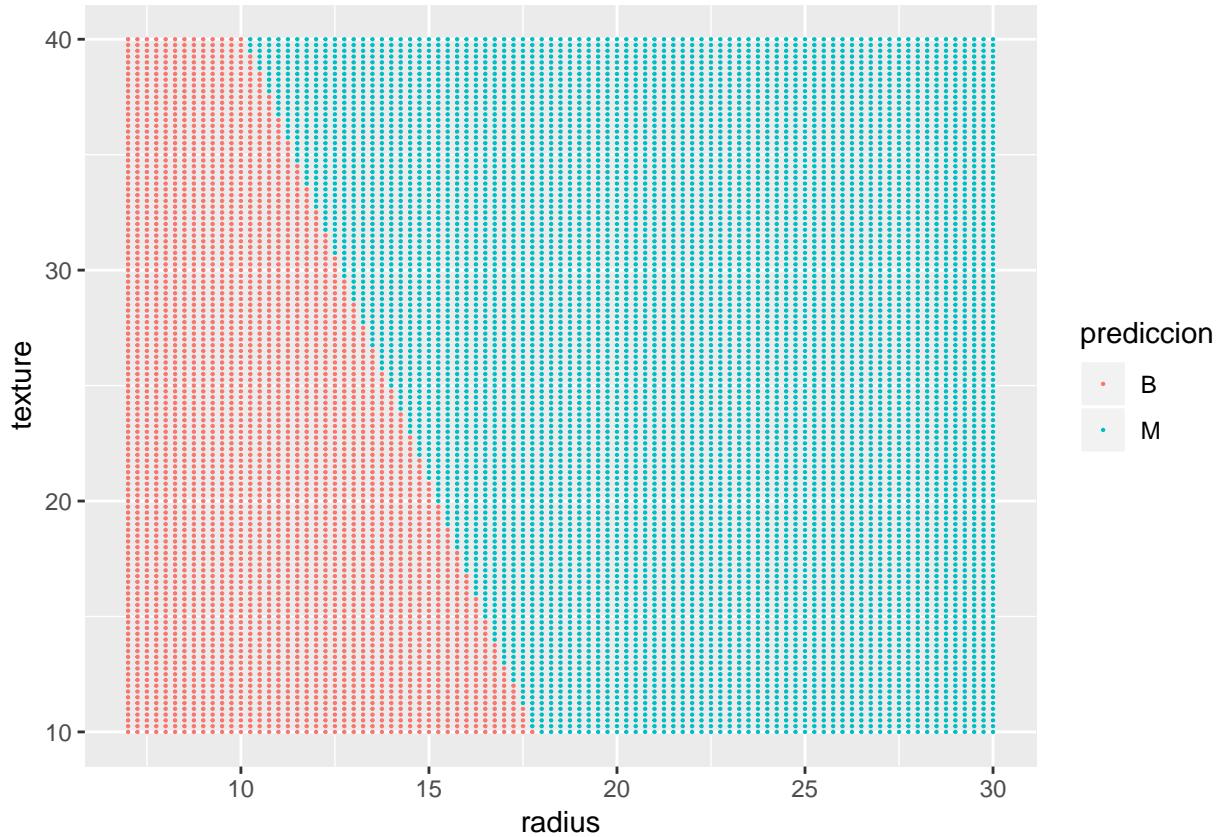
A continuación aplicamos este enfoque a los datos sobre el cáncer de mama, donde gracias al paquete caret los cambios en el código son mínimos:

```
##### Análisis Lineal Discriminante
train.lda <- train(diagnosis ~ radius + texture, data = Train, method = "lda")
predicciones=predict(train.lda,Test)
confusionMatrix(predicciones, Test$diagnosis , positive="M")

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   B    M
##           B 167  19
##           M   8  79
##
##           Accuracy : 0.9011
##                 95% CI : (0.8594, 0.9338)
##     No Information Rate : 0.641
##     P-Value [Acc > NIR] : < 2e-16
##
##           Kappa : 0.7797
##   Mcnemar's Test P-Value : 0.05429
##
##           Sensitivity : 0.8061
##           Specificity : 0.9543
##     Pos Pred Value : 0.9080
##     Neg Pred Value : 0.8978
##           Prevalence : 0.3590
##           Detection Rate : 0.2894
##     Detection Prevalence : 0.3187
##     Balanced Accuracy : 0.8802
##
##     'Positive' Class : M
##
```

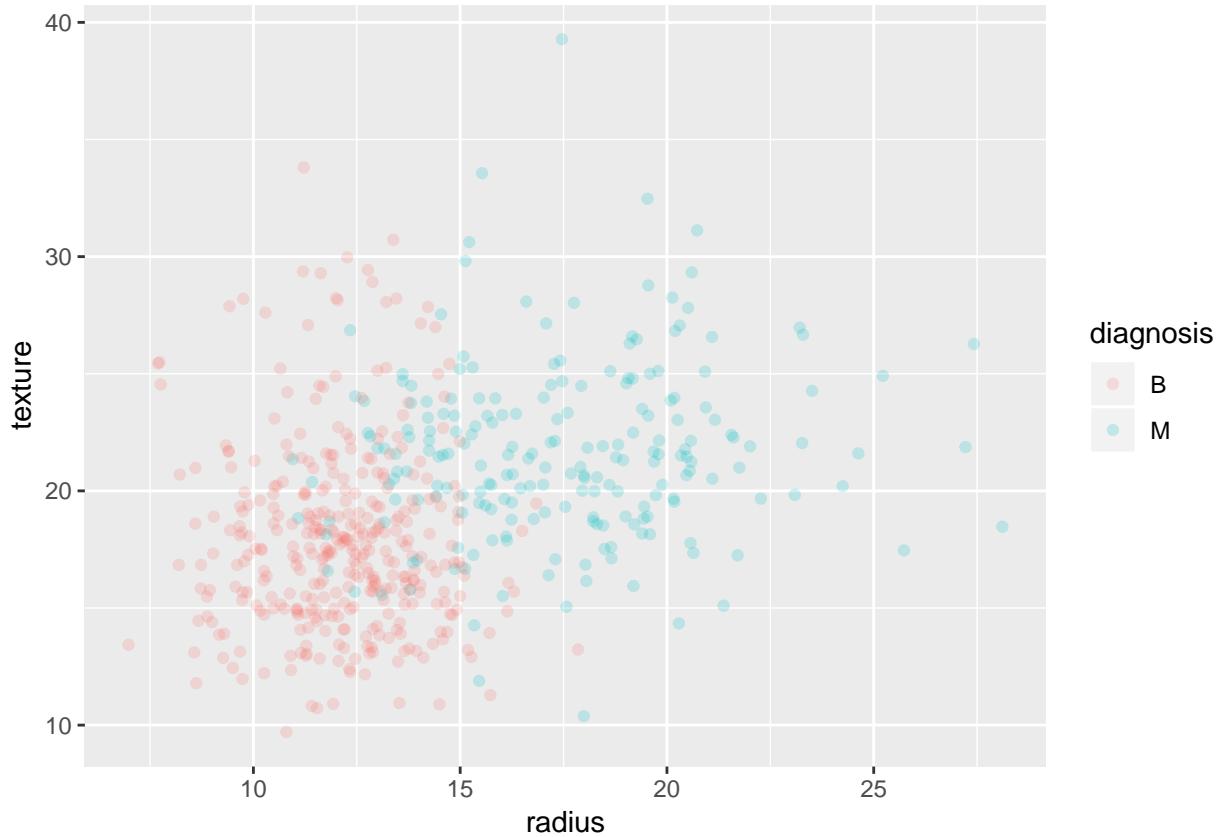
Cuyos resultados son globalmente similares a los obtenidos con la regresión logística.

```
##### Visualización de la frontera del clasificador
new$prediccion=predict(train.lda,new)
ggplot(new,aes(x=radius,y=texture,color=prediccion))+geom_point(size=.1)
```



Si aflojamos las asunciones del modelo de Análisis Lineal Discriminante dejando que la matriz de varianza de X pueda también cambiar de una clase a otra, obtenemos el modelo de *análisis discriminante cuadrático*. Esta hipótesis parece más razonable a la vista de los datos:

```
ggplot(mama,aes(x=radius,y=texture,color=diagnosis))+geom_point(alpha=.2)
```



A continuación evaluamos la precisión de este método sobre los datos de cáncer de mama.

```
#####
# Análisis Discriminante Cuadrático
train.qda <- train(diagnosis ~ radius + texture, data = Train, method = "qda")
predicciones=predict(train.qda,Test)
confusionMatrix(predicciones, Test$diagnosis , positive="M")

## Confusion Matrix and Statistics
##
##          Reference
## Prediction   B   M
##           B 164  16
##           M  11  82
##
##          Accuracy : 0.9011
## 95% CI : (0.8594, 0.9338)
##  No Information Rate : 0.641
## P-Value [Acc > NIR] : <2e-16
##
##          Kappa : 0.7827
##  Mcnemar's Test P-Value : 0.4414
##
##          Sensitivity : 0.8367
##          Specificity : 0.9371
##  Pos Pred Value : 0.8817
##  Neg Pred Value : 0.9111
##          Prevalence : 0.3590
```

```

##           Detection Rate : 0.3004
##     Detection Prevalence : 0.3407
##     Balanced Accuracy : 0.8869
##
##           'Positive' Class : M
##

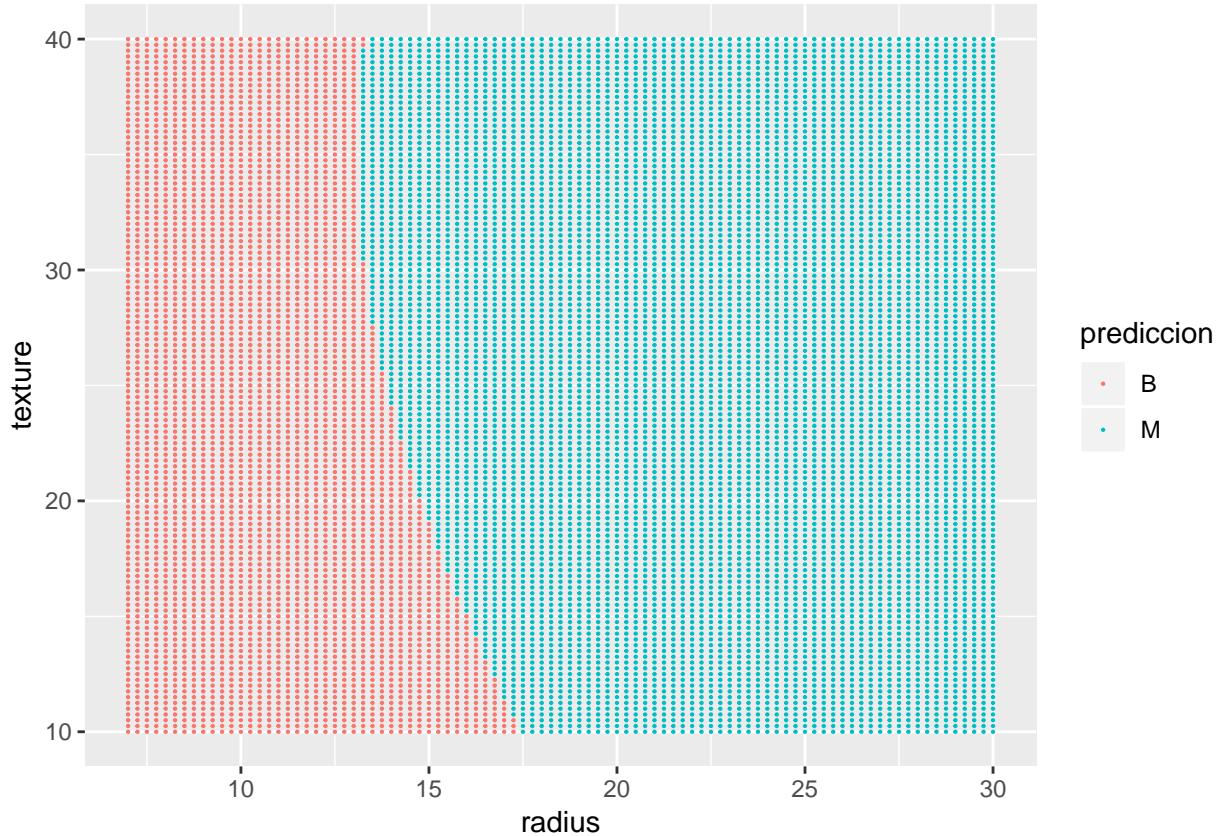
```

Que da resultados mejores.

```

##### Visualización de la frontera del clasificador
new$prediccion=predict(train.qda,new)
ggplot(new,aes(x=radius,y=texture,color=prediccion))+geom_point(size=.1)

```



3.6. K-vecinos más cercanos

vamos ahora nuestro primer método de clasificación *no paramétrico* : el método de k -vecinos más cercanos. Hasta ahora, todos los métodos de clasificación que hemos visto han sido paramétricos. Por ejemplo, la regresión logística tenía la forma:

$$\log \left(\frac{p(x)}{1 - p(x)} \right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p.$$

En este caso, los parámetros β_j del modelo que hemos estimado (“aprendido”) ajustando (“adiestrando”) el modelo. Luego, estas estimaciones permitieron estimar la probabilidad $p(x) = P(Y = 1 | X = x)$:

$$\hat{p}(x) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p}}$$

El método de k -vecinos no tienen tales parámetros. En su lugar, tiene un * parámetro de ajuste* k que determina la complejidad del modelo y que podemos seleccionar de acuerdo al desempeño del clasificador fuera de la muestra.

Si a veces se ve el método de k -vecinos como una “caja negra” que devuelve un clasificador, es en realidad un método de aproximación no paramétrico de las probabilidades $p_c(x) = P(Y = c | X = x)$:

$$\hat{p}_c(x) = \hat{P}(Y = c | X = x) = \frac{1}{g} \sum_{i \in \mathcal{N}_k(x)} (y_i = c)$$

Es decir, la proporción de clase g entre los k vecinos de x . Luego, como es habitual, asignamos a x la clase que maximiza esta probabilidad.

$$\hat{y}_x = \operatorname{argmax}_{c \in 1, 2, \dots, C} \hat{p}_c(x)$$

En otras palabras, asignamos la clase que más voto acumula en la vecindad de x . Si dos clases tienen la misma probabilidad se asigna una de las dos al azar.

En el caso de una variable dicotómica ($C = 2$), esto se convierte en

$$\hat{y}_x = \begin{cases} 1 & \hat{p}_g(x) > 0,5 \\ 0 & \hat{p}_g(x) \leq 0,5 \end{cases}$$

```
# Define el grid de parámetros a probar
valores <- expand.grid(k = seq(5, 100, 5))

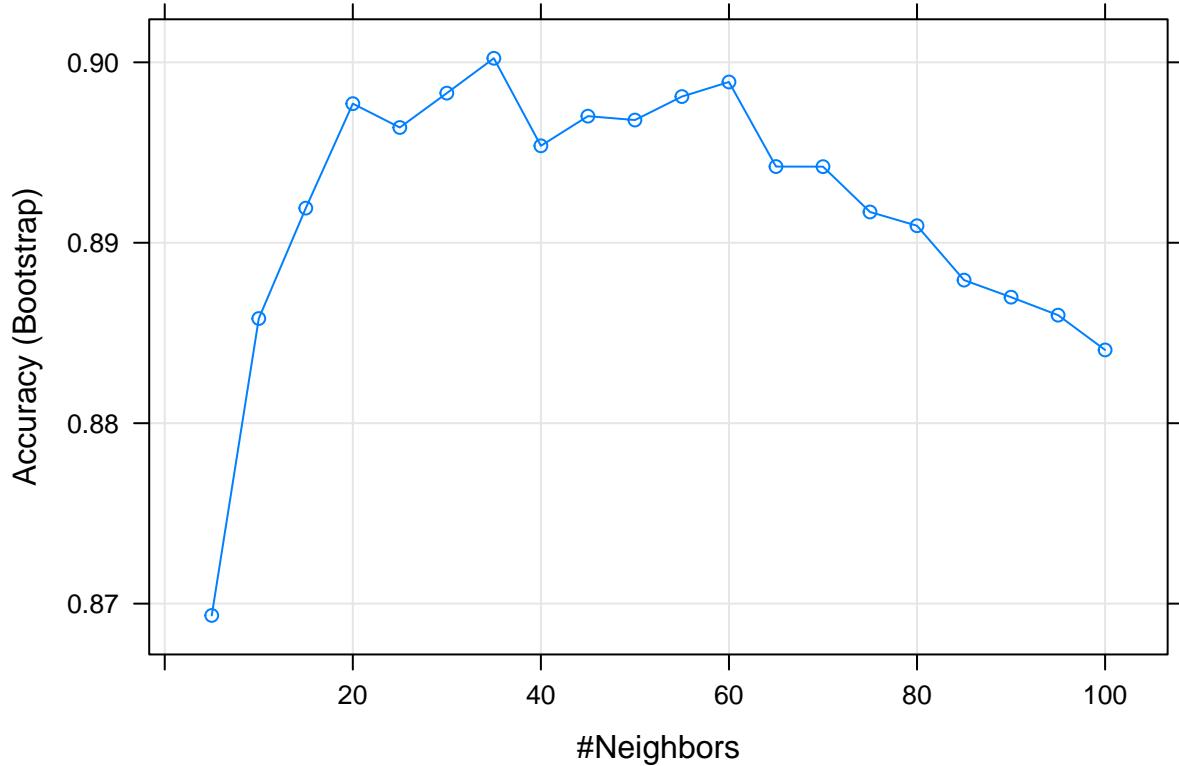
# Aplica el método seleccionando el valor óptimo de k
train.knn <- train(diagnosis ~ radius+texture, data = mama, method = 'knn', tuneGrid = valores)
train.knn

## k-Nearest Neighbors
##
## 569 samples
##    2 predictor
##    2 classes: 'B', 'M'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 569, 569, 569, 569, 569, 569, ...
## Resampling results across tuning parameters:
##
##     k     Accuracy   Kappa
##     5    0.8693460  0.7151827
##    10   0.8858004  0.7494502
##    15   0.8919197  0.7627934
##    20   0.8977086  0.7762309
##    25   0.8963884  0.7733214
##    30   0.8982927  0.7773941
##    35   0.9002238  0.7815019
##    40   0.8953720  0.7708711
##    45   0.8970162  0.7741769
##    50   0.8968010  0.7733182
##    55   0.8981018  0.7763439
```

```

##      60  0.8989107  0.7780681
##      65  0.8942234  0.7682070
##      70  0.8942183  0.7682350
##      75  0.8917064  0.7622614
##      80  0.8909438  0.7605055
##      85  0.8879305  0.7535433
##      90  0.8869841  0.7509258
##      95  0.8859878  0.7485995
##     100  0.8840613  0.7443244
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 35.
plot(train.knn)

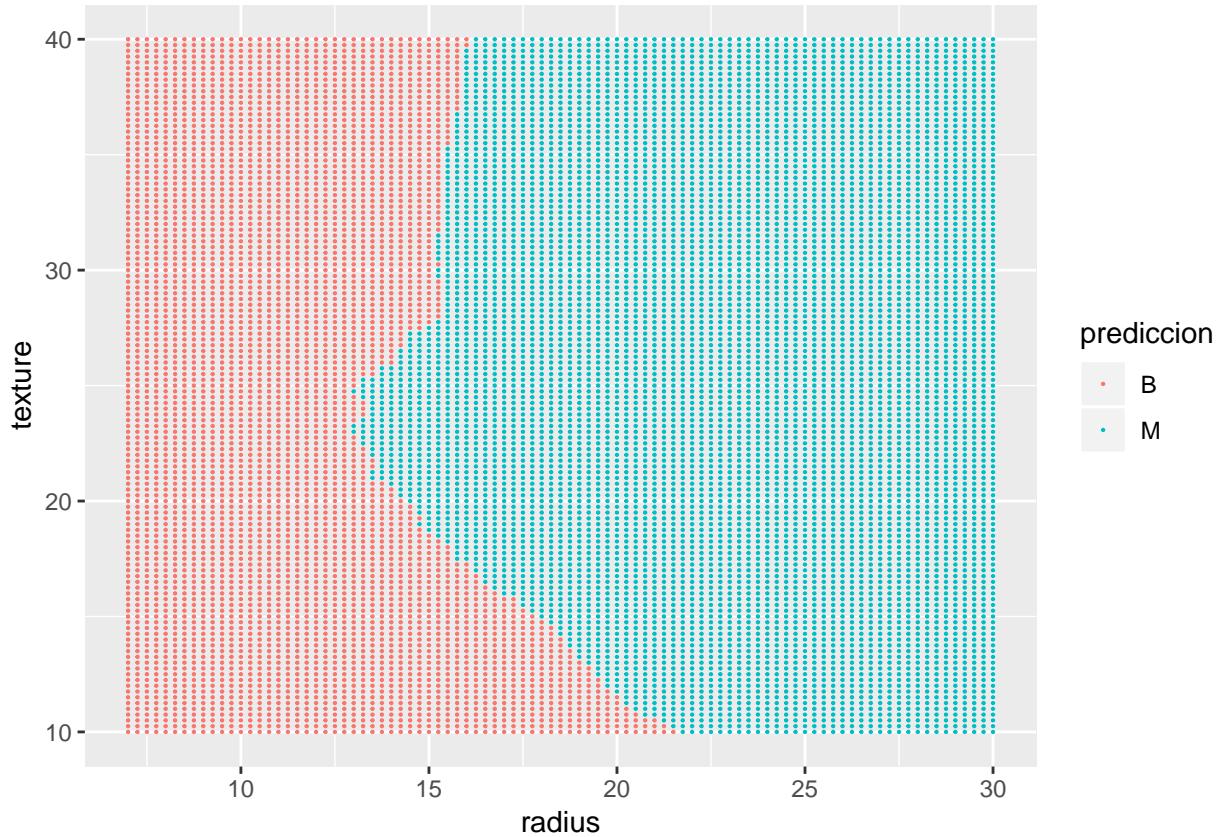
```



```

##### Visualización de la frontera del clasificador
new$prediccion=predict(train.knn,new)
ggplot(new,aes(x=radius,y=texture,color=prediccion))+geom_point(size=.1)

```



Ejercicio 3.3 Comparar los distintos métodos de clasificación a la predicción del movimiento del índice de S&P.

4. Regularización

4.1. Preámbulo

Cuando se utiliza la regularización?:

- Problema de predicción con un alto número de predictores (overfitting)
- Mucho ruido en la base de datos (muchos predictores no están relacionados con la respuesta)
- Alta correlación entre los predictores (multicolinealidad)

Porqué Regularizar?

- La estimación de los coeficientes puede ser muy inestable (altas varianzas)
- Permite una selección *continua* de variables
- Subsanar los problemas que conlleva la multicolinealidad

En qué consiste la regularización?

Regresión lineal:

Para $i = 1, 2, \dots, N$,

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_K x_{iK} + \varepsilon_i$$

donde $\varepsilon_i \sim N(0, \sigma^2)$.

los coeficientes β_k se estiman sin restricción.

Regresión con regularización:

los coeficientes β_k están restringido.

1. *Ridge regression*:

$$\sum_{k=1}^K \beta_k^2 \leq t$$

2. *Lasso regression*:

$$\sum_{k=1}^K |\beta_k| \leq t$$

3. *Elastic-net*: mixtura de Ridge y Lasso

$$\sum_{k=1}^K [\alpha |\beta_k| + (1 - \alpha) \beta_k^2] \leq t$$

Interpretación de la regularización:

- **Shrinkage**:

- Ridge: asemeja los coeficientes de los predictores mutuamente correlacionados
- Lasso: selecciona uno de estos predictores y descarta los demás.

- **PCA**: Penaliza las direcciones (combinaciones) de los predictores poco informativas.

- **Bayesian**:

- Ridge \sim los coeficientes tienen una distribución (*a priori*) gaussiana.
- Lasso \sim los coeficientes tienen una distribución (*a priori*) de Laplace.

4.2. Regularización con el paquete glmnet

```
require(glmnet)
```

Un ejemplo de trenes sencillo para empezar

```
##### Retrasos de trenes #####
set.seed(4321) #fija la semilla para que el ejemplo sea reproducible.
n=1000 #tamaño de la muestra
DT=data_frame(distancia=rnorm(n,500,100))
DT <- DT %>% mutate(estaciones = rpois(n,10), precio:=rnorm(n,50,10))
DT=round(DT,2)
DT

## # A tibble: 1,000 x 3
##       distancia   estaciones     precio
##           <dbl>        <dbl>      <dbl>
## 1        457.         16     45.9
## 2        478.         8      57.8
## 3        572.         13     48.6
## 4        584.         10     59.1
## 5        487.          6     52.9
## 6        661.          9     55.2
## 7        470.         14     51.6
## 8        520.          9     65.6
## 9        624.          7     38.9
```

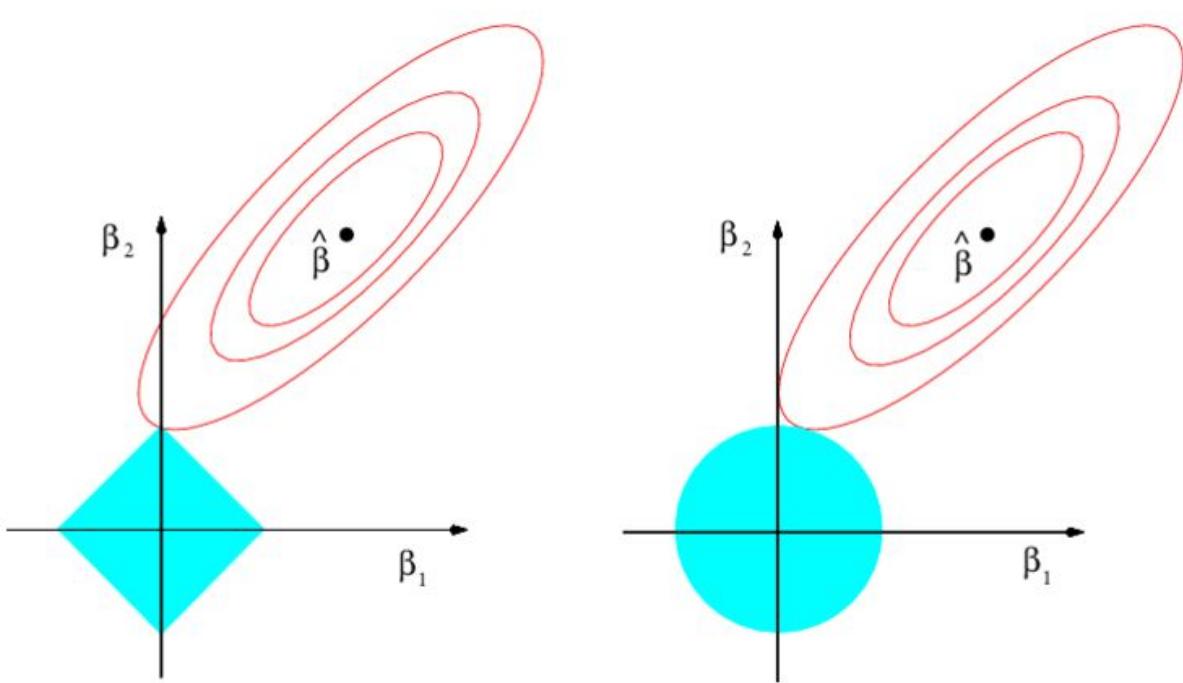
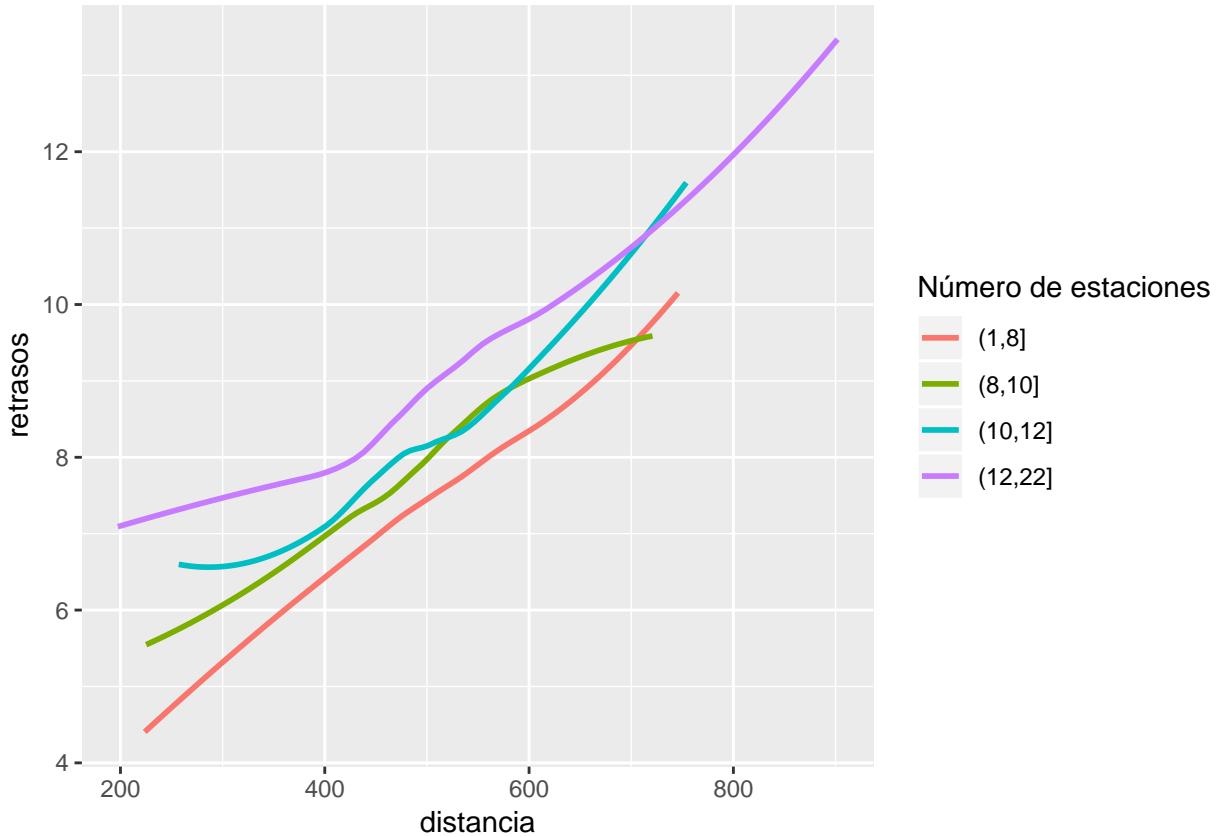


Figura 3: Contornos del elipse de confianza de los parametros y regiones de restriccción (en azul) para la regularización de Lasso (izquierda) y Ridge (derecha).

```
## 10      428.      12    37.2
## # ... with 990 more rows
```

Ahora generamos los retrasos

```
DT <- DT %>% mutate(retrasos:= 1 + distancia/100 + estaciones/5 + rnorm(n,0,1) )
ggplot(DT,aes(y=retrasos,x=distancia,color=cut(estaciones,quantile(estaciones),right=TRUE))) +
  geom_smooth(se=FALSE) + scale_color_discrete("Número de estaciones")
```



Resultado de una estimación lineal

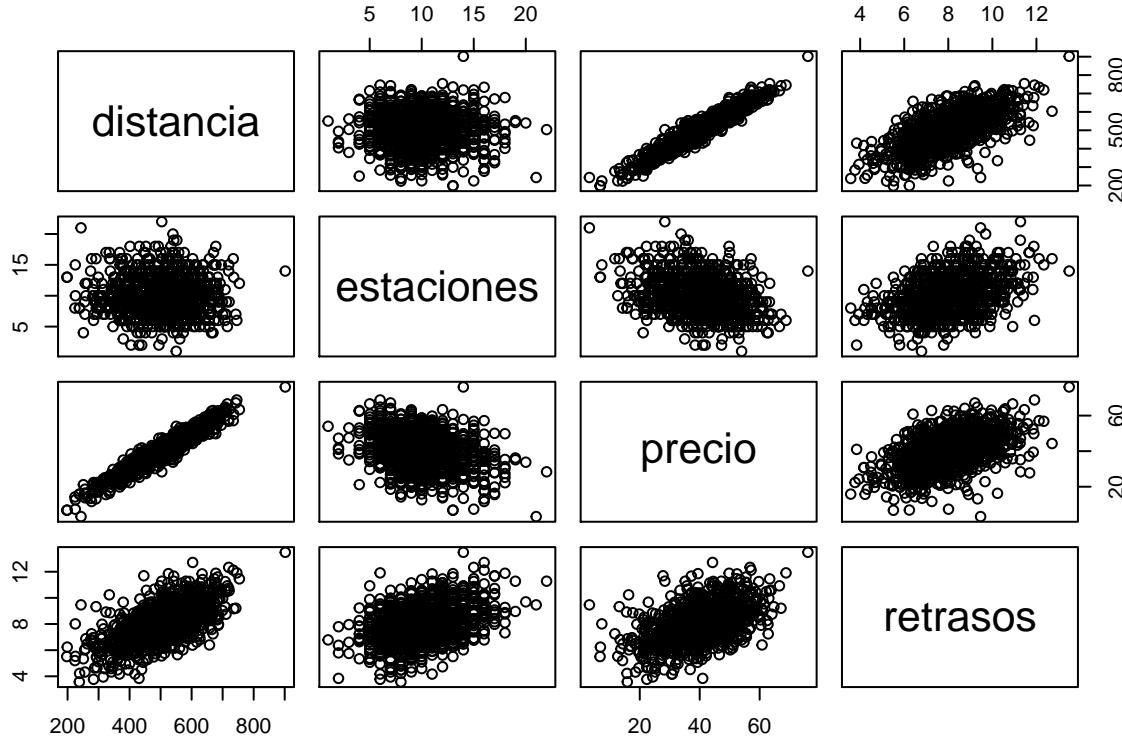
```
ols <- lm(retrasos ~ . ,DT)
summary(ols)

##
## Call:
## lm(formula = retrasos ~ ., data = DT)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -3.2259 -0.6230  0.0089  0.6358  3.0901 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 1.2131745  0.2470188  4.911 1.06e-06 ***
## distancia    0.0096490  0.0003088 31.248  < 2e-16 ***
## estaciones   0.1870312  0.0100462 18.617  < 2e-16 ***
## precio       0.0020255  0.0031895  0.635     0.526    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9704 on 996 degrees of freedom
## Multiple R-squared:  0.5681, Adjusted R-squared:  0.5668 
## F-statistic: 436.6 on 3 and 996 DF,  p-value: < 2.2e-16
```

La estimación es bastante buena!

Pero, en realidad el precio está muy correlacionado con los demás predictores

```
DT <- DT %>% mutate(precio = distancia/10 - estaciones + rnorm(n,0,.1) ) #precio y distancia están corr  
pairs(DT)
```



Ajustamos de nuevo una regresión lineal:

```
ols <- lm(retrasos ~ . ,DT)  
summary(ols) #Nada es significativo!!!
```

```
##  
## Call:  
## lm(formula = retrasos ~ ., data = DT)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max  
## -3.2243 -0.6226  0.0042  0.6238  3.0971  
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 1.31601    0.18772   7.010 4.38e-12 ***  
## distancia    0.01297    0.03182   0.408   0.684  
## estaciones   0.15342    0.31879   0.481   0.630  
## precio      -0.03316   0.31824  -0.104   0.917  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Residual standard error: 0.9705 on 996 degrees of freedom
```

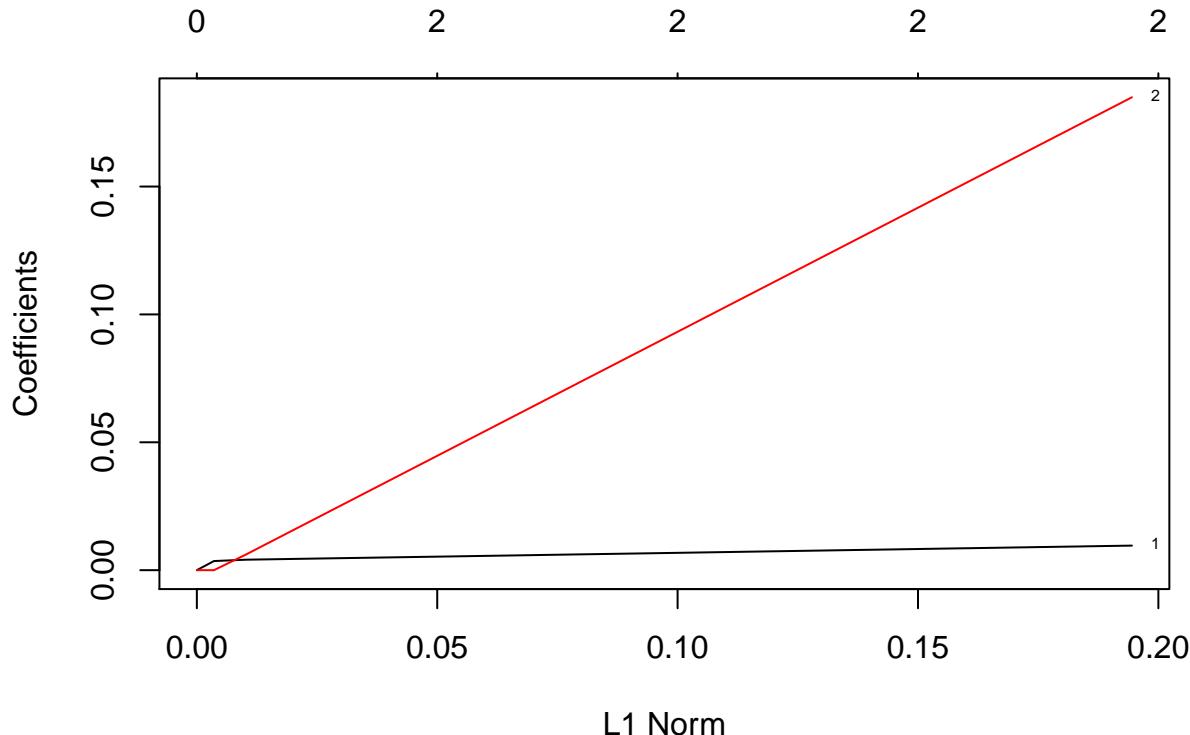
```
## Multiple R-squared:  0.5679, Adjusted R-squared:  0.5666
## F-statistic: 436.3 on 3 and 996 DF,  p-value: < 2.2e-16
```

Para solucionar este problema utilizamos una regresión de Lasso via glmnet:

```
y=DT$retrasos
x=subset(DT,select=-retrasos)
x

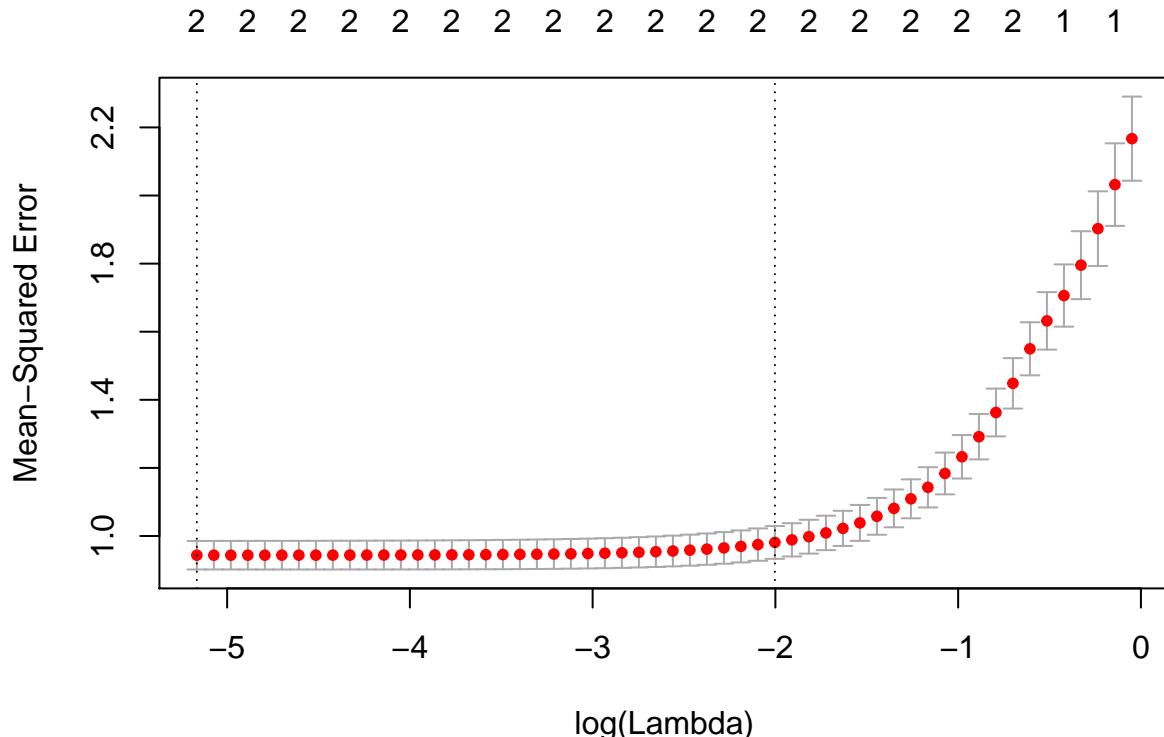
## # A tibble: 1,000 x 3
##       distancia estaciones precio
##          <dbl>      <dbl>   <dbl>
## 1        457.        16    29.7
## 2        478.        8     40.0
## 3        572.        13    44.1
## 4        584.        10    48.5
## 5        487.        6     42.8
## 6        661.        9     57.1
## 7        470.        14    33.0
## 8        520.        9     42.9
## 9        624.        7     55.5
## 10       428.       12    30.9
## # ... with 990 more rows

lasso=glmnet(as.matrix(x),y)
plot(lasso,label=TRUE)
```



Para seleccionar el coeficiente de la penalización recurrimos a cross-validation:

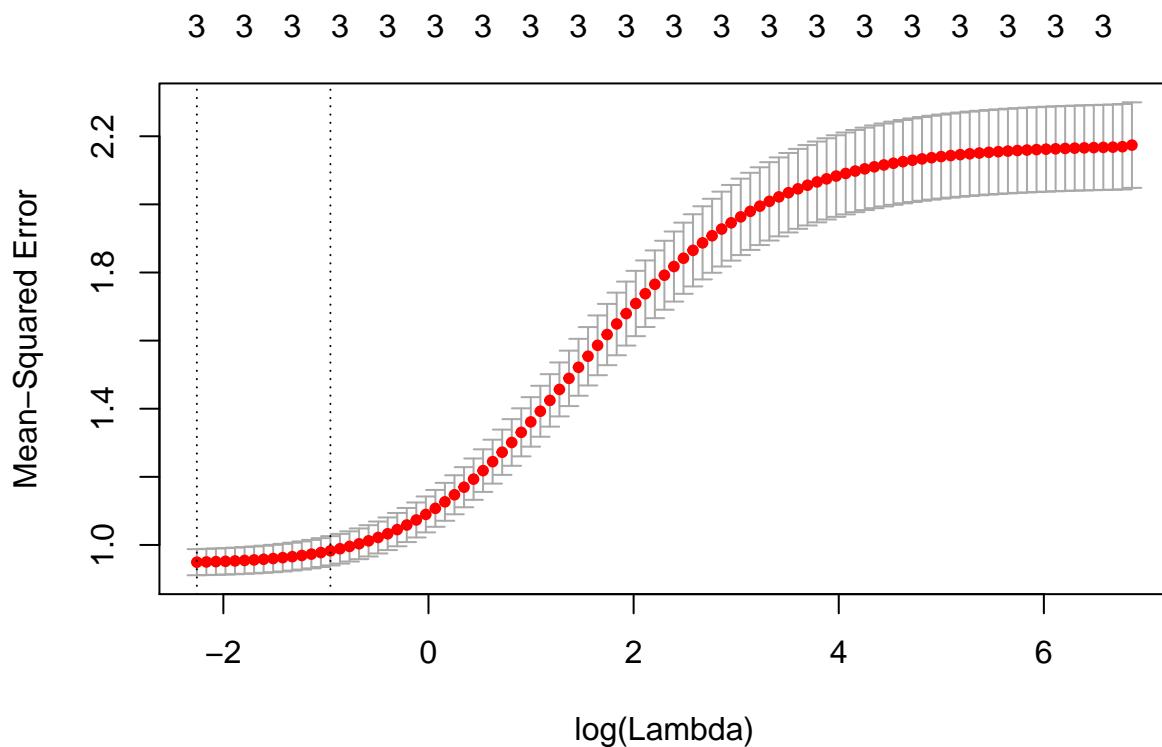
```
cvlasso = cv.glmnet(as.matrix(x), y)
plot(cvlasso)
```



```
coef(cvlasso, s = "lambda.min")  
## 4 x 1 sparse Matrix of class "dgCMatrix"  
## (Intercept) 1  
## distancia 0.009596977  
## estaciones 0.184735707  
## precio .
```

Qué haría la regresión de Ridge?

```
cvridge = cv.glmnet(as.matrix(x), y, alpha=0)
plot(cvridge)
```



```

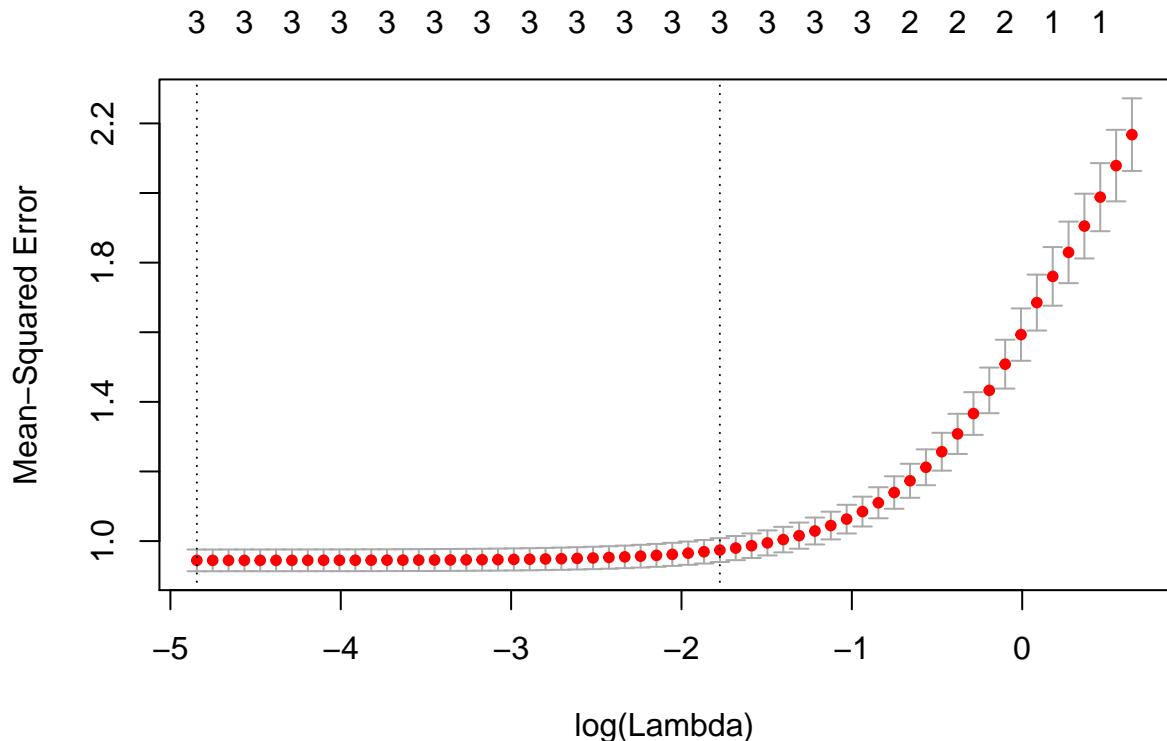
coef(cvr ridge, s = "lambda.min")

## 4 x 1 sparse Matrix of class "dgCMatrix"
##           1
## (Intercept) 1.670022808
## distancia   0.005803715
## estaciones  0.206200389
## precio      0.034372194

Y con la elastic-net

cvnEN = cv.glmnet(as.matrix(x), y, alpha=.5)
plot(cvnEN)

```



```

coef(cvnEN, s = "lambda.min")

## 4 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept) 1.364829073
## distancia    0.008572283
## estaciones   0.195003272
## precio       0.010201757

```

La multicolinealidad afecta la estimación de los coeficientes de la regresión, pero no tiene porque afectar la predicción.

4.2.1. Overfitting

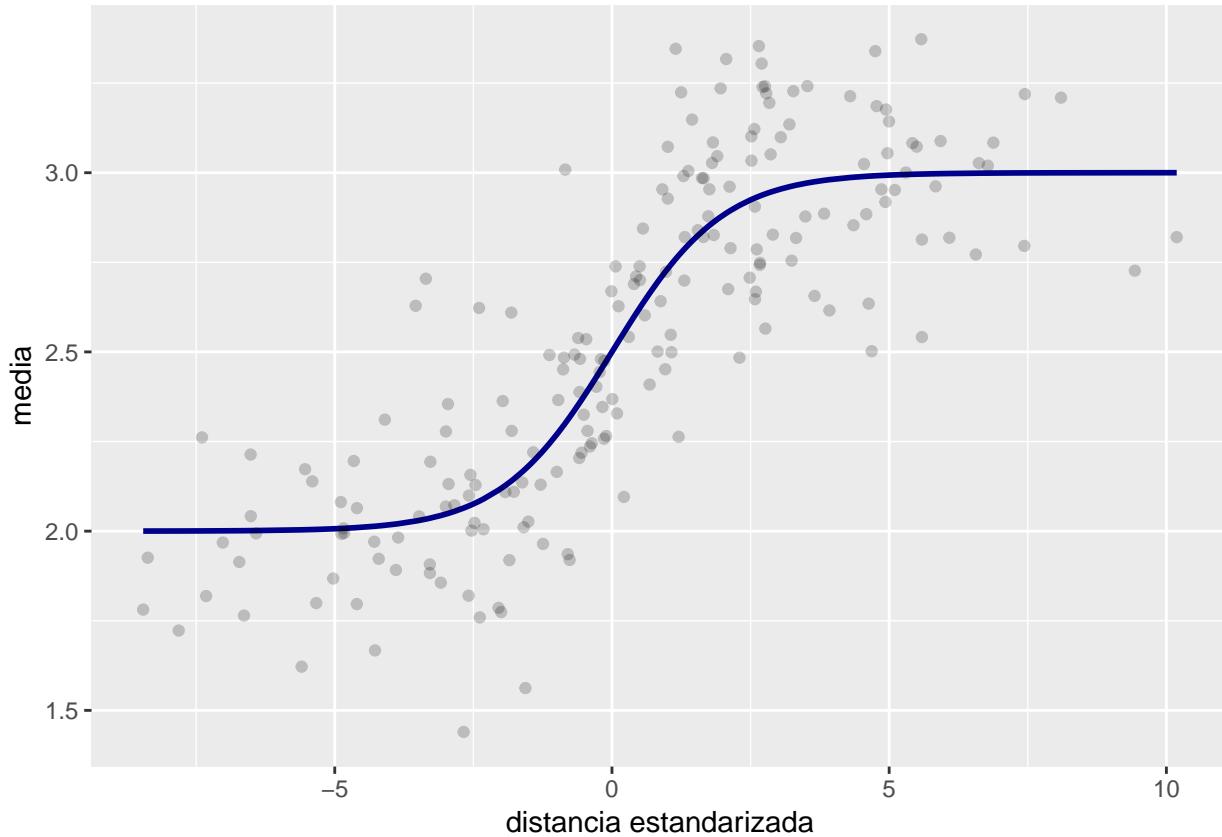
Para ilustrar este problema, seguimos con el ejemplo de los trenes.

```

n=200
DT=data_frame(distancia=rnorm(n,0,4)) #distancias estandarizadas
DT<- DT %>% mutate(media = 2 + exp(distancia)/(1+exp(distancia)))
DT<- DT %>% mutate(retrasos = media + rnorm(n,0,.25) )

p<-ggplot(DT,aes(y=media,x=distancia)) + geom_line(size=1,color="blue4") +
  ylim(range(DT$retrasos))+xlab("distancia estandarizada")
p + geom_point(aes(y=retrasos),alpha=.2)

```



Para investigar la capacidad predictiva del modelo, partimos la base de datos en un conjunto de adiestramiento y un conjunto de test:

```
set.seed(4321)
index = sample(c(TRUE, FALSE), nrow(DT), replace=TRUE, prob=c(.5, .5))
Train = DT[index, ]
Test = DT[!index, ]
```

Queremos ajustar una regresión con funciones suaves (polinomios)

```
ols <- lm(retrasos ~ poly(distancia, 10) , data=Train)
summary(ols)
```

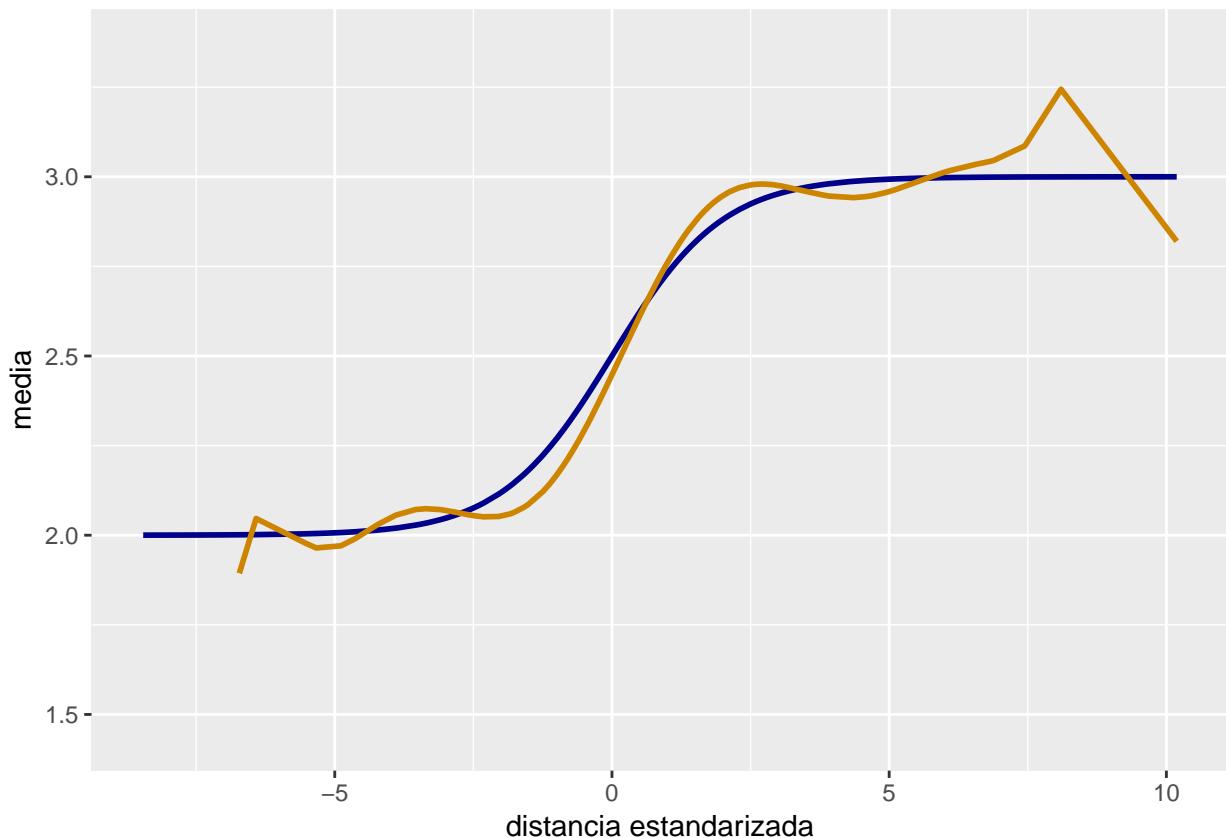
```
##
## Call:
## lm(formula = retrasos ~ poly(distancia, 10), data = Train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.61980 -0.14067 -0.02577  0.13512  0.63021 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 2.5857     0.0226 114.421 < 2e-16 ***
## poly(distancia, 10)1 3.7136     0.2348 15.813 < 2e-16 ***
## poly(distancia, 10)2 -0.7231     0.2348 -3.079  0.00270 ** 
## poly(distancia, 10)3 -1.1646     0.2348 -4.959 3.01e-06 ***
## poly(distancia, 10)4  0.5979     0.2348  2.546  0.01247 *  
##
```

```

## poly(distancia, 10)5    0.3815    0.2348   1.624  0.10753
## poly(distancia, 10)6   -0.7570    0.2348  -3.224  0.00173 **
## poly(distancia, 10)7   -0.3863    0.2348  -1.645  0.10324
## poly(distancia, 10)8    0.3010    0.2348   1.282  0.20301
## poly(distancia, 10)9    0.1034    0.2348   0.440  0.66075
## poly(distancia, 10)10   -0.3122    0.2348  -1.330  0.18678
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2348 on 97 degrees of freedom
## Multiple R-squared:  0.7616, Adjusted R-squared:  0.7371
## F-statistic: 30.99 on 10 and 97 DF,  p-value: < 2.2e-16
#overfitting !!
Train$fit.ols=ols$fit

p<- p + geom_line(aes(x=distancia,y=fit.ols),data=Train,col="orange3",size=1)
p

```



```

pred=predict(ols,Test)
mse=mean((pred-Test$retrasos)^2)
sprintf("MSE_ols = %.3f",mse)

```

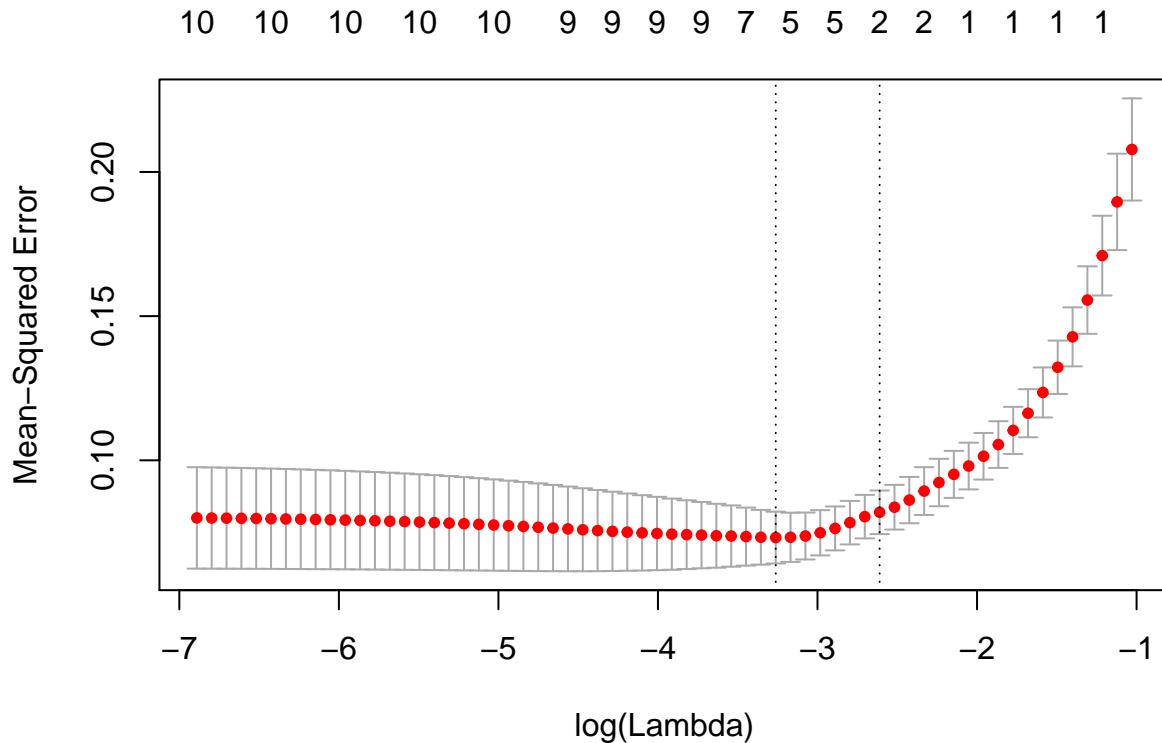
```
## [1] "MSE_ols = 8.636"
```

Para regularizar esta regresión, utilizamos de nuevo glmnet

```

x=model.matrix(retrasos ~ poly(distancia,10) , data=Train)[,-1]
y=Train$retrasos
lasso = cv.glmnet(as.matrix(x), y)
plot(lasso)

```



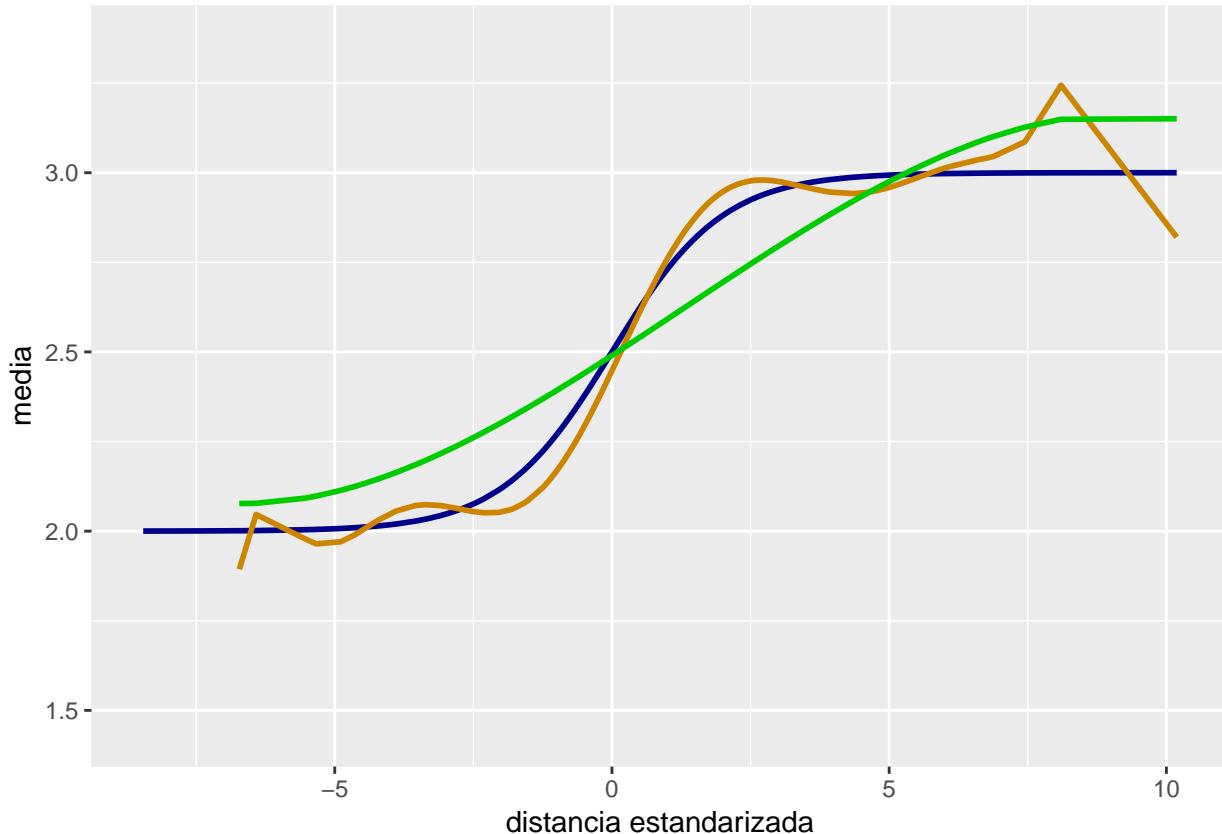
```

coef(lasso, s = "lambda.1se")

## 11 x 1 sparse Matrix of class "dgCMatrix"
##                               1
## (Intercept)      2.5856664
## poly(distancia, 10)1 2.9498846
## poly(distancia, 10)2 .
## poly(distancia, 10)3 -0.4008951
## poly(distancia, 10)4 .
## poly(distancia, 10)5 .
## poly(distancia, 10)6 .
## poly(distancia, 10)7 .
## poly(distancia, 10)8 .
## poly(distancia, 10)9 .
## poly(distancia, 10)10 .

Y predecimos
#####
# Ajuste regularizado
fit.glmnet=predict(lasso,newx=as.matrix(x),s = "lambda.1se")
Train$fit.glmnet=fit.glmnet
p + geom_line(aes(x=distancia,y=fit.glmnet),data=Train,col="green3",size=1)

```



```
#####
## Calculo del error de validación
x.new=model.matrix(retrasos ~ poly(distancia,10) , data=Test)[,-1]
pred=predict(lasso,newx=as.matrix(x.new),s = "lambda.min")
mse=mean((pred-Test$retrasos)^2)
sprintf("MSE_glmnet = %.3f",mse)

## [1] "MSE_glmnet = 0.077"
```

4.2.2. Ejercicios: Tres contextos de Big Data

Determinar la regularización correcta en los tres contextos siguientes:

4.2.2.1. Contexto 1:

Verdadero modelo: $y = X\beta + \epsilon$

donde

- $\beta = (\underbrace{1, \dots, 1}_{15}, \underbrace{0, \dots, 0}_{4985})$
- $p = 5000 > n = 1000$
- Predictores no correlacionados:

$$X_i \stackrel{\text{iid}}{\sim} N(0, I)$$

- $\epsilon \stackrel{\text{iid}}{\sim} N(0, I)$

```
set.seed(1234)
```

```
n <- 1000 # Número de observación
```

```

p <- 5000 # Numero de predictores
p0 <- 15 # Numero de predictores correlacionados con la respuesta
DT1 <- as_data_frame(matrix(rnorm(n*p), nrow=n, ncol=p))
names(DT1)<-sub("V","X",names(DT1))
DT1 <- DT1 %>% mutate(y=rowSums(.[1:p0]) + rnorm(n)) # y = X1+X2+...+X15 + eps

# partición de la muestra en train (2/3) y test (1/3)
DT1 <- DT1 %>% mutate(muestra=sample(c("train","test"),n,prob=c(2/3,1/3),replace=TRUE))
subset(DT1,select=c(y,X1,X2,muestra))

## # A tibble: 1,000 x 4
##       y     X1     X2 muestra
##   <dbl> <dbl> <dbl> <chr>
## 1 -5.57 -1.21 -1.21 test 
## 2  2.61  0.277  0.301 test 
## 3 -6.47  1.08 -1.54 train 
## 4  3.65 -2.35  0.635 test 
## 5  1.15  0.429  0.703 train 
## 6 -2.95  0.506 -1.91 train 
## 7 -3.75 -0.575  0.939 train 
## 8  0.206 -0.547 -0.224 test 
## 9 -0.343 -0.564 -0.674 train 
## 10 -5.65 -0.890  0.446 test 
## # ... with 990 more rows

```

4.2.2.2. Contexto 2:

Verdadero modelo: $y = X\beta + \epsilon$

donde

- $\beta = (\underbrace{1, \dots, 1}_{1500}, \underbrace{0, \dots, 0}_{3500})$
- $p = 5000, n = 1000$
- Predictores no correlacionados:

$$X_i \stackrel{\text{iid}}{\sim} N(0, I)$$

- $\epsilon \stackrel{\text{iid}}{\sim} N(0, I)$

```

set.seed(1234)
n <- 1000 # Numero de observación
p <- 5000 # Numero de predictores
p0 <- 1500 # Numero de predictores correlacionados con la respuesta
DT2 <- as_data_frame(matrix(rnorm(n*p), nrow=n, ncol=p))
names(DT2)<-sub("V","X",names(DT2))
DT2 <- DT2 %>% mutate(y=rowSums(.[1:p0])+rnorm(n)) # y = X1+X2+...+X1500 + eps

# partición de la muestra en train (2/3) y test (1/3)
DT2 <- DT2 %>% mutate(muestra=sample(c("train","test"),n,prob=c(2/3,1/3),replace=TRUE))
subset(DT2,select=c(y,X1,X2,muestra))

```

```

## # A tibble: 1,000 x 4
##       y     X1     X2 muestra
##   <dbl> <dbl> <dbl> <chr>
## 1  3.55 -1.21 -1.21 test 
## 2 48.9   0.277  0.301 test 

```

```

## 3 -18.5   1.08  -1.54  train
## 4 66.7   -2.35   0.635 test
## 5 -48.5    0.429   0.703 train
## 6 18.4    0.506  -1.91  train
## 7 7.45   -0.575   0.939 train
## 8 41.9   -0.547  -0.224 test
## 9 -50.2   -0.564  -0.674 train
## 10 -10.1   -0.890   0.446 test
## # ... with 990 more rows

```

4.2.2.3. Contexto 3:

Verdadero modelo: $y = X\beta + \epsilon$

donde

- $\beta = (10, 10, 5, 5, \underbrace{1, \dots, 1}_{10}, \underbrace{0, \dots, 0}_{36})^T$
- $p = 50$
- $n = 100$
- Predictores correlacionados:

$$\text{Cov}(X)_{ij} = (0,9)^{|i-j|}$$

```

set.seed(1234)
require(MASS)
n <- 100 # Número de observación
p <- 50 # Número de predictores
Covarianza <- outer(1:p, 1:p, function(x,y) {0.9^abs(x-y)})
DT3 <- as_data_frame(mvrnorm(n, rep(0,p), Covarianza))
names(DT3)<-sub("V","X",names(DT3))
DT3 <- DT3 %>% mutate(y=rowSums(.[5:14])) # y = X5+X6+...+X14
DT3 <- DT3 %>% mutate(y= y + 10 * (X1+X2) + 5*(X3+X4)+rnorm(n)) # y = 10*(X1+X2) + 5*(X3+X4) + X5+X6+...
# partición de la muestra en train (2/3) y test (1/3)
DT3 <- DT3 %>% mutate(muestra=sample(c("train","test"),n,prob=c(2/3,1/3),replace=TRUE))
subset(DT3,select=c(y,X1,X2,muestra))

## # A tibble: 100 x 4
##       y      X1      X2 muestra
##   <dbl>  <dbl>  <dbl> <chr>
## 1  2.12   1.29   0.574 test
## 2 19.8    0.710   0.471 train
## 3  8.43  -0.208  -0.129 train
## 4 -47.2   -1.08  -0.898 train
## 5  25.8    0.827   0.880 train
## 6 -18.5   -1.25  -0.857 train
## 7 -25.4   -0.135  -1.24  train
## 8 -33.0   -0.739  -0.514 train
## 9 -11.7   -0.559  -0.441 train
## 10 -0.270  0.0764 -0.0281 train
## # ... with 90 more rows

```

j Evaluar los distintos tipos de regularizaciones Lasso, Ridge y elastic-net) en estos tres contextos. Para ello se podrá utilizar la función MSE que viene a continuación

```

MSE<-function(DT,output="y",alpha=1){#
  Train=subset(DT,muestra=="train",select=-muestra)#

```

```

Test=subset(DT,muestra="test",select=-muestra)#
Y=as.matrix(Train[,output]) #output#
X=as.matrix(subset(Train,select=-get(output))) #inputs: todo salvo output#
X.new=as.matrix(subset(Test,select=-get(output))) # new inputs#
fit=cv.glmnet(X,Y,alpha=alpha)#
pred=predict(fit,newx=X.new,s="lambda.min")#
Y.new=as.matrix(Test[,output])
mean((Y.new-pred)^2) # MSE#
}

```

4.2.3. Clasificación

No hay novedad, salvo que ahora la respuesta es dicotómica.

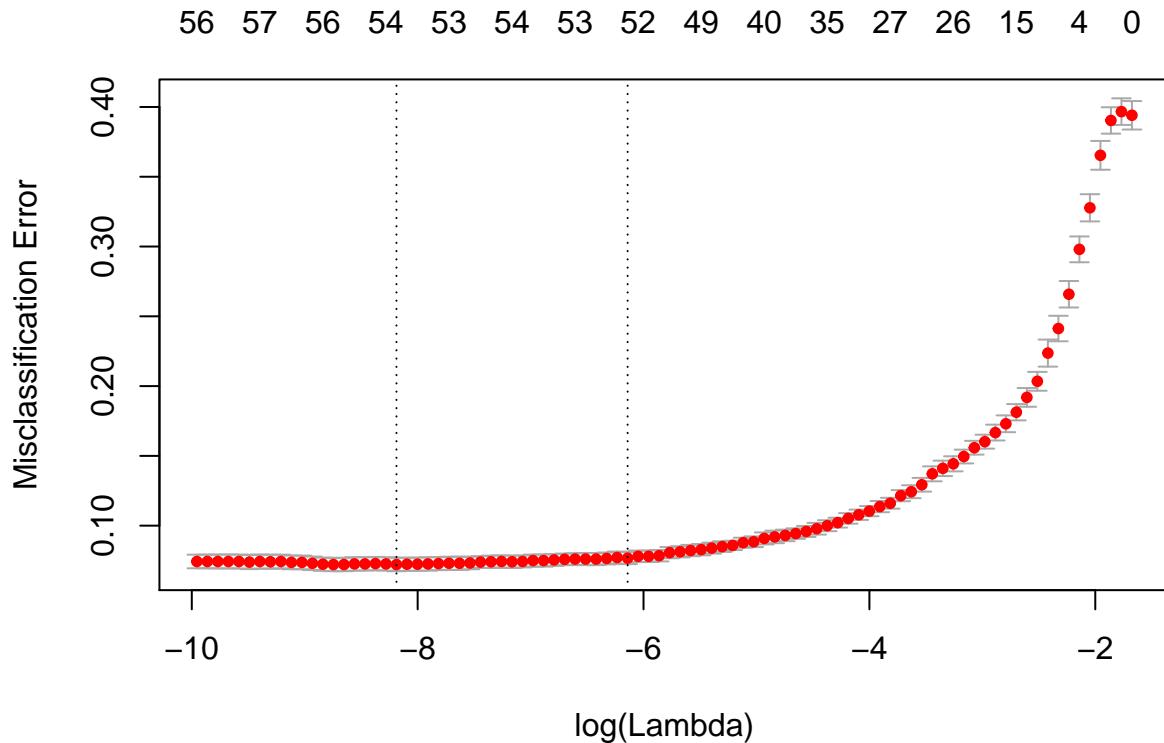
Cargamos un ejemplo:

```
spam=read.csv("data/spam.csv")  
glimpse(spam)
```

```
## $ WF85 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ WFtechnology <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WF1999 <dbl> 0.00, 0.07, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WFparts <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ WFpm <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ WFdirect <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WFcs <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ WFmeeting <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ WForiginal <dbl> 0.00, 0.00, 0.12, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WFproject <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WFre <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WFEdu <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00, 0.00, 0.00,...  
## $ WFTable <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ WFconference <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...  
## $ CFSemicolon <dbl> 0.000, 0.000, 0.010, 0.000, 0.000, 0.000, 0.000,...  
## $ CFOpenParenthesis <dbl> 0.000, 0.132, 0.143, 0.137, 0.135, 0.223, 0....  
## $ CFOpenBracket <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0....  
## $ CFMark <dbl> 0.778, 0.372, 0.276, 0.137, 0.135, 0.000, 0....  
## $ CFDollar <dbl> 0.000, 0.180, 0.184, 0.000, 0.000, 0.000, 0....  
## $ CFSsharp <dbl> 0.000, 0.048, 0.010, 0.000, 0.000, 0.000, 0....  
## $ CAPave <dbl> 3.756, 5.114, 9.821, 3.537, 3.537, 3.000, 1....  
## $ CAPlon <int> 61, 101, 485, 40, 40, 15, 4, 11, 445, 43, 6,...  
## $ CAPtot <int> 278, 1028, 2259, 191, 191, 54, 112, 49, 1257...  
## $ Status <fct> Spam, Spam, Spam, Spam, Spam, Spam, Sp...
```

Ajuste con glmnet

```
y=spam$Status  
x=as.matrix(subset(spam,select=-c(Status)))  
cvfit = cv.glmnet(x, y, family = "binomial", type.measure = "class")  
plot(cvfit)
```



Para evaluar la tasa de acierto en una muestra de validación, podemos utilizar la siguiente función:

```
spam$muestra=sample(c("train","test"),nrow(spam),replace=TRUE,prob=c(2/3,1/3))
ROC<-function(DT,output="Status",alpha=1){#
  Train=subset(DT,muestra=="train",select=-muestra)#
  Test=subset(DT,muestra=="test",select=-muestra)#
  Y=Train[,output] #output#
  X=as.matrix(subset(Train,select=-get(output))) #inputs: todo salvo output#
  X.new=as.matrix(subset(Test,select=-get(output))) # new inputs#
  fit=cv.glmnet(X,Y,alpha=alpha,family = "binomial", type.measure = "class")#
  pred=predict(fit,newx=X.new,s="lambda.min",type="class")#
  Y.new=Test[,output]#
  confusionMatrix(table(pred,Y.new)) # tabla de predicciones versus realidad#
}
ROC(spam,alpha=1)
```

4.2.4. Ejercicios: Análisis de datos reales

Repositorio de la UCI:

<https://archive.ics.uci.edu/ml/datasets.html>

Ejemplo “Calidad del vino”

```
#require(data.table)
#vinos=fread("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv")
#dim(vinos)
#x=as.matrix(subset(vinos,select=-quality))
```

```
#y=vinos$quality  
#fit = glmnet(x, y, family = "multinomial", type.multinomial = "grouped")  
#plot(fit, label=TRUE)
```

Otros conjuntos de datos:

```
# - https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset (regresión)  
# - https://archive.ics.uci.edu/ml/datasets/Bank+Marketing (clasificación)  
# - https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients (clasificación)
```